(http://csmatters.org) 4 - 7

0b100 - 0b111

Hypothesis Testing with Simulations in NetLogo

Unit 4. Data Acquisition

Revision Date: Jul 22, 2019
Duration: 3 50-minute sessions

Lesson Summary

Summary

This lesson teaches students to use simulations to develop, refine and test hypotheses. NetLogo, which is used throughout the lesson to illustrate the use of functional and data abstraction, is a programmable modeling environment for simulating natural and social phenomena.

NetLogo is a variation of the Logo language instead of Python, so students are not expected to write new code in this lesson. See http://www.ianbicking.org/docs/PyLogo_lightning.html (http://www.ianbicking.org/docs/PyLogo_lightning.html) for a comparison of Logo and Python.

Outcomes

- Students will understand that models are abstractions of real environments and will recognize the rationale for, and limitations of, modeling techniques to analyze problems.
- Students will be able to compare abstractions in one programming language to another (Python vs NetLogo)
- Students will recognize the use of functional and data abstractions in modeling.
- Students will be able to develop and test hypotheses using an experimental approach in a modeling framework.

Overview

Session 1 - Modeling in NetLogo

- Getting Started (8 min)
- Learning NetLogo (40 min)
- Wrap Up (2 min)

Session 2 - Models and Hypothesis Design

- Getting Started (5 min)
- Models and Hypotheses (20 min)

- Model Selection and Hypothesis Development (25 min)

Session 3 - Hypothesis Testing

- Getting Started (5 min)
- Hypothesis Testing (40 min)
- Wrap Up (5 min)

Note: This lesson introduces another programming tool and environment: NetLogo. Teachers may choose to complete only the first session (on the basics of NetLogo), to expose students to a new computational platform and way of thinking, to extend the ideas in Unit 4 about modeling and simulation.

Learning Objectives

CSP Objectives

- *EU CRD-1 - Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.*
    - LO CRD-1.A - Explain how computing innovations are improved through collaboration.
    - LO CRD-1.C - Demonstrate effective interpersonal skills during collaboration.
- *EU CRD-2 - Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.*
    - LO CRD-2.B - Explain how a program or code segment functions.
- *EU DAT-1 - The way a computer represents data internally is different from the way the data is interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.*
    - LO DAT-1.A - Explain how data can be represented using bits.
- *EU DAT-2 - Programs can be used to process data, which allows users to discover information and create new knowledge.*
    - LO DAT-2.D - Extract information from data using a program.
    - LO DAT-2.E - Explain how programs can be used to gain insight and knowledge from data.
- *EU AAP-1 - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.*
    - LO AAP-1.B - Determine the value of a variable as a result of an assignment.
- *EU AAP-2 - The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.*
    - LO AAP-2.A - Express an algorithm that uses sequencing without using a programming language.
    - LO AAP-2.B - Represent a step-by-step algorithmic process using sequential code statements.

- LO AAP-2.C - Evaluate expressions that use arithmetic operators.
- LO AAP-2.H - For selection: a. Write conditional statements. b. Determine the result of conditional statements.
- LO AAP-2.K - For iteration: a. Write iteration statements. b. Determine the result or side-effect of iteration statements.

- *EU AAP-3 - Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.*
  - LO AAP-3.A - For procedure calls: a. Write statements to call procedures. b. Determine the result or effect of a procedure call.
  - LO AAP-3.B - Explain how the use of procedural abstraction manages complexity in a program.
  - LO AAP-3.D - Select appropriate libraries or existing code segments to use in creating new programs.
  - LO AAP-3.E - For generating random values: a. Write expressions to generate possible values. b. Evaluate expressions to determine the possible results.
  - LO AAP-3.F - For simulations: a. Explain how computers can be used to represent real-world phenomena or outcomes. b. Compare simulations with real-world contexts.

- *EU IOC-1 - While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.*
  - LO IOC-1.A - Explain how an effect of a computing innovation can be both beneficial and harmful.

Key Concepts

Students will understand that models are an abstraction of real environments and will recognize the rationale for and limitations of modeling techniques to analyze problems.

Students will recognize the use of functional and data abstractions in modeling.

Students will be able to develop and test hypotheses using an experimental approach in a modeling framework.

Essential Questions

- How can computing extend traditional forms of human expression and experience?
- How does abstraction help us in writing programs, creating computational artifacts and solving problems?
- How can computational models and simulations help generate new understanding and knowledge?
- How are algorithms implemented and executed on computers and computational devices?
- Why are some languages better than others when used to implement algorithms?
- How are programs developed to help people, organizations or society solve problems?
- How are programs used for creative expression, to satisfy personal curiosity or to create new knowledge?

- How do computer programs implement algorithms?
- How does abstraction make the development of computer programs possible?
- Which mathematical and logical concepts are fundamental to computer programming?

Teacher Resources

Student computer usage for this lesson is: **required**

NetLogo (http://ccl.northwestern.edu/netlogo/). http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL. NetLogo tutorial packet online web version http://ccl.northwestern.edu/netlogo/docs/ (http://ccl.northwestern.edu/netlogo/docs/)

Modeling and Simulation 101 video (https://www.youtube.com/watch?v=X-6zxImekOE (https://www.youtube.com/watch?v=X-6zxImekOE))

See http://www.ianbicking.org/docs/PyLogo_lightning.html (http://www.ianbicking.org/docs/PyLogo_lightning.html) for a comparison of Logo, PyLogo and Python.

New Mexico "Computer Science for All" bases the entire course on modeling and simulation using NetLogo http://www.cs4all.org/NM-CS108L-Week3-Final (http://www.cs4all.org/NM-CS108L-Week3-Final)

Lesson Plan

Session One

Getting Started (8 min)

Question: Describe something that can be modeled using a simulation.

Introduce modeling and simulation using the first four minutes of the Modeling and Simulation 101 video (https://www.youtube.com/watch?v=X-6zxImekOE) (https://www.youtube.com/watch?v=X-6zxImekOE (https://www.youtube.com/watch?v=X-6zxImekOE)). Students open a document for notes for today's session.

Students should record and briefly discuss these four statements about modeling and simulation:

1. Models describe real-world activities using abstration, enabling testing of hypotheses at a fraction of the cost of actual experiments.
2. Models can be expressed in computational, mathematical, textual, and/or graphical forms.
3. Simulations are forms or instances of models that can be implemented as computer programs.
4. Monte Carlo simulations exploit randomness to arrive at their results.

Learning NetLogo (40 min)

To start, all students should download NetLogo from this link http://ccl.northwestern.edu/netlogo/ (http://ccl.northwestern.edu/netlogo/) or use the web version of the program.

Students should work through the NetLogo tutorial packet (http://ccl.northwestern.edu/netlogo/docs/) either in groups or as a class.

In particular, students should be encouraged to notice commonalities in programming languages (sequence, conditionals, iteration, abstraction) and how differences in languages provide specific tools best suited to particular problems. The domain of modeling and simulation is a huge area in computational thinking, and NetLogo is one of many languages well suited to problem-solving in this domain. Point out that Python is used for modeling and simulation, but to be clear and readable requires the abstraction of libraries to build on that provide the same functionality that comes with a language like NetLogo.

There will be some "thought questions" throughout that students should discuss in their groups and as a class.

Wrap Up (2 min)

Students should complete an exit ticket listing one interesting idea they learned, or one question they have about NetLogo or modeling.

Session Two

Getting Started (5 min)

Review yesterday's NetLogo lesson and ask the students to share what they learned, how NetLogo is similar to or different from Python, and any questions they have about how it works. Point out that it is the ABSTRACTION available in NetLogo that makes it easier to read and write simulation programs because it has features built into it that are readily available that you can build on. This PowerPoint details the abstractions in NetLogo (http://www.cs4all.org/files/CS108L_Week3_Abstraction_in_Life.pptx) with the accompanying transcript (http://www.cs4all.org/files/Abstraction%20V1.docx) from CS for All in New Mexico. List abstractions available in Python that are different from NetLogo or PyLogo. What is built into each language that makes it easy to use?

Models and Hypotheses (20 min)

Introduction (12 min)

a. Students start NetLogo and open the Art>Fireworks model.
b. Students use the interface Buttons for Setup and Go to run the simulation.
c. Have students read the information in the Info tab, and look through the code in the Code tab, to get a sense of what is being simulated and how it works.
d. Explain that the **model** is the *description* of the **environment,** while the **simulation** is the *specific implementation* of the **model**.

e. **Think -Pair- Share:** Ask the students to identify which aspects of the real-world environment (actual fireworks) ***are*** implemented in the model, and which aspects ***are* not** implemented.
f. Discuss with the class the fact that the simpler characteristics of the model or abstraction make the implementation of the model much easier, at the expense of a possible failure to represent key relationships.
g. Have the students examine the implementation (Code tab) and find the names of the five functions defined in the code. Show where all 5 are called (three in the code and two in the Interface).
h. Discuss these facts about forming a hypothesis
   - A hypothesis is an educated guess about how things work.
   - A hypothesis is written like this: "If _____[I do] _____, then _____[this will result]_____
i. Explore hypotheses that could be tested by this model. (Some hypotheses are suggested by the discussion in the Info tab -- e.g., "If gravity is set to 0, then ______."
j. Ask:
   - What are two things that we can change through the interface?
   - What do we think will happen we we make those changes?

Activity (8 min)

Ask each student to write a hypothesis that can be tested with this simulation, share the hypothesis with elbow partners, and briefly experiment with the parameters to informally test the hypothesis.

Model Selection and Hypothesis Development (25 min)

Note: The "Hypothesis Testing Worksheet" which will be used for the next two lessons is available in the Lesson Resources Folder

For the rest of today's session and Session 3, students will work in teams of four students to select a model to experiment with, then divide into two partner sets to develop a hypothesis, devise an experimental plan, test the hypothesis, and write about their results.

Directions

1. First divide the class into teams of four students, and have them spend 10 minutes exploring different models in NetLogo (which they should already have some familiarity with from Session 1 and the earlier part of Session 2). They should choose one model to focus on and write the name of the model in the corresponding part of the Hypothesis Testing worksheet, along with a short explanation of why they chose the model.
2. For the next 10 minutes, they should study the parameters of the model and how it works, trying things out in the interface to see how changing the parameters affects behavior.
3. In the final 5 minutes of the class session, split the teams into partner groups and have each partner group begin to develop hypotheses about the team's selected model. You may wish to assign them to write these hypotheses down as a homework assignment, either together or individually, using the Hypothesis Testing worksheet to record their hypothesis, why they selected that hypothesis, and the experimental parameters they will use to test the hypothesis.

Session Three

Warm Up (5 min)

Partners should revisit their hypotheses, and choose one hypothesis to focus on first. (They can test both hypotheses if they have time.) Each partner pair should write the name of their model and their selected hypothesis on the board, to share with the other students.

Experimental Design (10 min)

1. Students should use the Hypothesis Testing worksheet to:
    a. Identify an experiment that will test their hypothesis.
    b. Select one or more parameters to vary.
    c. Choose what settings they will use for these parameters.
    d. Determine what measurements they will record.
    e. Decide how many trials they will run at each setting.
    f. **Note:** Be sure that students have completed their experimental design ***before*** proceeding to perform the experiment -- many students will want to just jump in and start trying things, but one goal of the lesson is to help them understand that having a clear hypothesis and experimental design before collecting data is an important part of the scientific process.

Hypothesis Testing (30 min)

For the next twenty to thirty minutes, students should carry out their experiments and record the appropriate measurements.

At the end of the section or for homework, students should write up their findings in a short report, showing the data they've collected (optionally in a graphical form, particularly if assigned as homework), discussing what the data says about their hypothesis, and concluding whether the hypothesis is supported or refuted by the simulation.

Wrap Up (5 min)

Students should come back into their teams to share their findings, and discuss the advantages and disadvantages of using models and simulations to develop and test hypotheses.

Evidence of Learning

Formative Assessment

Students will share and post their hypothesis before testing and sharing the results. Teachers will verify that the hypothesis are falsifiable and testable by the simulations.

Summative Assessment

Students will select a model, develop a hypothesis, design an experiment, and use a simulation to test the hypothesis.

(http://www.umbc.edu/) (http://www.umd.edu/)

(http://www.nsf.gov/)

Authored by: CS Matters in Maryland
Website: csmatters.org (http://csmatters.org)
Email: csmattersinmaryland@gmail.com (mailto:csmattersinmaryland@gmail.com)