CHAPTER

6

Coding in Science

When integration talk turns to science courses, I'm often asked by teachers and administrators whether computer science is a "science" in the same way we think of chemistry, biology, or physics as sciences. It's a reasonable question, given that "science" is right there in the name. If we do think of computer science as a domain of science equal to Earth, life, or physical science, it might change our approach to finding space for it in the school day. While the National Science Teachers Association (NSTA) hasn't released an official position statement articulating the role of computer science in science education, NSTA Executive Director David Evans published a statement titled "Computer Science Should Supplement, not Supplant Science Education" to the NSTA blog (Evans, 2016). The statement was released in response to a number of related events, including the release of the K–12 Computer Science Framework, an increasing number of states allowing computer science to count for math or science graduation credit, and California Governor Jerry Brown signing a bill to teach CS in every grade. In his statement, Evans expresses concern about the impact of increased computer science adoption on science classes, with a particular focus on the fear that computer science courses may supplant traditional science courses in high schools, to the detriment of students.

The Argument for Coding in Science

While I am inclined to argue against some of Evans' positions, such as his defeatist assumption that there's no time in the day to teach new subjects, he does make good points pertinent to our goals here. First, he points out that there are principles of computer science included in the NGSS:

> Computer Science principles can be found in the NGSS. Science and Engineering Practices include developing and using models, and using mathematics and computational thinking. In the integrated STEM classroom, using the principles of NGSS, educators are working to seek out real-world, relevant, authentic problems that would be of interest to students and ask them to apply computational thinking to solve the problem using data analysis, visualization, seeking patterns, and computation. (Evans, 2016)

In support of this overlap, or perhaps even interdependence, between computer science and traditionally recognized sciences, Evans suggests that in K–8 we teach computer science within the context of existing math and science courses. The authors of the K–12 Computer Science Framework have attempted to articulate this overlap, and opportunity for integration, with a Venn diagram (See Figure 6.1). The overlapping standards are shared in Table 6.1.

None of this actually clarifies whether computer science *is* a science. Peter Denning (2005) explored this question from an industry perspective. He proposed that the two are inherently interconnected, if technically independent. While many in the computer science and science communities have argued for computer science to be considered a first-class object within the NGSS, the reality is that it is—by design—a supporting character (Bienkowski, 2015). For the purposes of integration into K–12 curriculum, I propose that we consider computer science plays a role similar to engineering in the NGSS. It's a set of skills and practices that are essential to the study of all sciences (to varying degrees), and a set of discrete skills and knowledge that should be taught alongside other sciences. While the NGSS may not consider computer science a first-class object that warrants its own science course, it *is* more than merely a tool for teaching the subjects. This is the viewpoint from which I'll explore approaches to integration.

Major Areas of Overlap

Looking at the K–12 Computer Science Framework's Venn diagram (see Figure 6.1), notice there are two major themes common to computer science and the NGSS: modeling and simulation, and data analysis. Sheena Vaidyanathan dove into

this overlap in an article for EdSurge, in which she explored potential paths for teaching that mirror my own experiences (Vaidyanathan, 2017). When it comes to modeling and simulation, the NGSS explicitly calls out the need for students to not only *use* computer models to explore scientific phenomena but also to modify and develop those models on their own (National Research Council, 2013). Irene Lee has lead this charge, developing excellent resources to support this overlap in her Project Growing Up Thinking Scientifically (GUTS) program, which was created as an after-school program and grew into a classroom-implemented module called "Computer Science in Science," in collaboration with Code.org (visit the Project GUTS website at **projectguts.com**).

In Project GUTS, students follow a trajectory called use-modify-create with a number of different multiagent models to explore everything from chemical reactions, to predator–prey relationships, to the spread of disease. This process first engages students in using an existing computational model to explore a scientific phenomenon. Second, they explore the code that runs the model to understand the assumptions made. Once students understand the model as provided, they can make simple modifications to the code to add new behaviors or better address simplifying assumptions. Finally, students are able to create their own models, using those they've seen before as a guideline. This approach encourages students to critically question other scientific models they encounter as those models move from unassailable black boxes to comprehendible, and modifiable, abstractions with necessary simplifications.

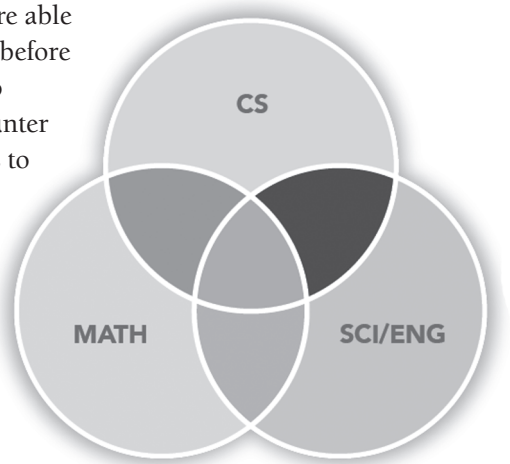

Figure 6.1: Computer science Venn diagram.

In one of my favorite progressions, students use a provided model to study the relationship between mountain lions and rabbits. By analyzing the code that drives this model, students learn that there are many missing pieces that could impact the accuracy of the model, such as the relationship between caloric intake and ability to move or reproduce.

The second major area of overlap, data, is a less trod ground by existing tools and curricula. The role that data plays in computing will be an increasingly important component of K–12 education, particularly because of the ways in which "big data" continues to change the world. As we develop better and more accessible tools for teaching students to grapple with big data, I expect we'll find many opportunities to bring the real world of modern science into the

Table 6.1 Overlap of NGSS Standards Between CS and Subject Areas

CS + MATH	CS + MATH + SCI/ENG	CS + SCI/ENG
DEVELOP AND USE ABSTRACTIONS M2. Reason abstractly and quantitatively M7. Look for and make use of structure M8. Look for and express regularity in repeated reasoning CS4. Developing and using abstractions **USE TOOLS WHEN COLLABORATING** M5. Use appropriate tools strategically CS2. Collaborating Around Computing **COMMUNICATE PRECISELY** M6. Attend to precision CS7. Communicating About Computing	**MODEL** S2. Develop and use models M4. Model with mathematics CS4. Developing and Using Abstractions CS6. Testing and Refining Computational Artifacts **USE COMPUTATIONAL THINKING** S5. Use mathematics and computational thinking CS3. Recognizing and Defining Computational Problems CS5. Creating Computational Artifacts **DEFINE PROBLEMS** S1. Ask questions and define problems M1. Make sense of problems and persevere in solving them CS3. Recognizing and Defining Computational Problems **COMMUNICATE RATIONALE** S7. Engage in argument from evidence S8. Obtain, evaluate, and communicate information M3. Construct viable arguments and critique the reasoning of others CS7. Communicating About Computing	**COMMUNICATE WITH DATA** S4. Anazlye and interpret data CS7. Communicating About Computing **CREATE ARTIFACTS** S3. Plan and carry out investigations S6. Construct explanations and design solutions CS4. Developing and Using Abstractions CS5. Creating Computational Artifacts CS6. Testing and Refining Computational Artifacts

Source: K–12 Computer Science Framework (2016)

classroom, recognizing that practicing scientists are increasingly spending more time analyzing the data from experiments than running those experiments, particularly as we find more ways for computers and robots to deal with the manual work of experimentation (Wolinsky, 2007).

That said, even the ability to collect and analyze small data computationally can present a significant hurdle. The aforementioned Project GUTS ties into this quite effectively, as the StarLogo Nova environment allows students to not only collect, but graph in real-time, data coming from their simulations. Alternatively, we can rely on curated data sets from companies, governments, and other organizations to provide students with real-world data. The Awesome Public Datasets project (**github.com/awesomedata/awesome-public-datasets**) provides access to hundreds of data sets, many freely available, organized by content type. Some curricula, such as Bootstrap's new Data Science course, are developing new tools and techniques to bring data science to new computer science students in ways that can be incorporated into the study of science, but also statistics, civics, or social studies (Krishnamurthi & Schanzer, 2017).

Unplugged Activities

Routing and Deadlock

Throughout their middle-school science classes, students learn about different systems, from the cellular level to the systems of planetary motion, and beyond. The NGSS defines systems as crosscutting concepts, meaning a system has "application across all domains of science" and is therefore "a way of linking the different domains of science" (National Research Council, 2013). Despite the apparent importance of teaching systems in all fields of science, and using them to help students draw connects between different domains, there is one incredibly ubiquitous system which is left out of the picture. A system that we engage with daily, even constantly. A system which, though man-made, exhibits many of the organic qualities of naturally occurring systems. Computer networks in general, and the internet in particular, are complex systems with myriad interesting behaviors to study and connect back to other systems in science. One such interesting behavior is the routing of information from one point to another.

The Orange Game is a CS Unplugged activity in which students participate in a group simulation of information traveling over a network (2002b). Students are seated in a circle and given a letter that serves as their address. Oranges, labeled with those same letters, are introduced to the circle. The network of students must

attempt to pass the oranges until all students have the orange(s) addressed to them. In the process, students will find that they have limited resources (hands), and if they attempt to work only in their own self-interest by holding onto the oranges addressed to them, those resources will become deadlocked and cannot be used to continue passing oranges to other students in the network. Though networks in real life are made of machines, they still need to balance the needs of the greater network with their own needs to ensure the whole system continues working.

Find the activity here: **creativecodingbook.com/unplugged/routing_and_deadlock**

Phylogenetics

The field of bioinformatics combines biology with computer science by using algorithms and data to solve problems in biology. One key advancement that bioinformatics has unlocked is the ability for biologists to reconstruct phylogenetic (evolutionary) trees, which can be used to trace how an animal's current genetic makeup evolved from an ancestor.

This CS Unplugged activity engages students in the process of reconstructing a phylogenetic tree using techniques from bioinformatics (2014). Though many of the concepts in this activity delve into topics typically introduced in higher-level math or science courses, the activity is designed for students as young as ten, and it can be used as a hands-on introduction to these concepts without requiring a great deal of prior knowledge. Using a list of words to stand in for the nucleotide bases, students play a game of "telephone,"—attempting to communicate a predetermined combination of words. Students record what they hear as they pass the message along, which leaves the class with record of how the message evolved through the game. By the end of the activity students will have used phylogenetics to reconstruct the evolutionary path of their game of telephone.

Find the activity here: **creativecodingbook.com/unplugged/phylogenetics**

CREATIVE CODING CONNECTIONS: **Project GUTS**

Project GUTS offers a middle school science program consisting of four instructional modules and professional development for the integration of computer science concepts into science classrooms through computer modeling and simulation.

projectguts.org

PROJECT: Lab Buddy

creativecodingbook.com/projects/lab_buddy

Overview

While we can certainly use App Lab to develop simple scientific models for study, it's not the ideal tool for developing models and simulations. Instead, rely on one of its cooler features—a simple-to-use database back end—to create a "lab buddy" app that mimics how real scientists might use programming to assist the gathering, processing, and display of data from a scientific experiment.

This project is a little bit different than the others, in that everyone uses the same app while gathering data from a science experiment. This is the most complicated program presented in this book, but students aren't required to do most (if any) of the programming. Borrowing from the use-modify-create method, students explore an example app to understand how it works before figuring out any necessary modifications for your chosen experiment. If modifications are necessary, they will be made by the teacher, with student assistance, beforehand.

Using this modified version of the sample app to record data, the class completes an experiment. Once the experiment is finished, they can export all of that data for basic analysis either in a separate App Lab app or a spreadsheet tool like Google Sheets or Microsoft Excel. Though the experiment won't produce big-data levels of data, it will certainly generate more than any one student would in an experiment, and it will be distributed in a fashion similar to how real big data is sourced!

Duration

If you do the entirety of this project, you should budget a day for students to get their hands dirty with App Lab, a day to run the experiment, and a day or two to analyze the data. To shorten the project, you can potentially eliminate the initial App Lab activity and have students use the shared app to gather data during the experiment, but then you lose any understanding of how the program functions or can be modified.

Objectives

- Gather a large volume of data with the aid of a computational tool.
- Summarize a large volume of data using a computational tool.

Vocabulary

Big Data: Extremely large data sets that, through computational analysis, can reveal trends and patterns.

Database: A structured set of data that can be written to, and read by, a program.

Object: In JavaScript, a type of variable that represents a collection of values.

Teacher Prep

The key to making this project effective is finding the right experiment to use for the data collection. You can use existing experiments you already teach, but you may need to reconsider the details to ensure that the type of data you're asking students to collect is both easy to collect in App Lab and lends itself to simple analysis through summarizing values. A good experiment for this might involve recording data that are:

- not dependent on time as factor in data collection (this makes it more difficult to summarize data across different students);
- easily categorized as a distinct set of values (e.g., the precipitate is red, blue, or green); or
- easily represented as a numeric value (e.g., temperature, pH, duration of reaction).

For the example app, I assume that students are measuring a single value across multiple specimens, such as the pH value of a series of liquids. You can increase the power and difficulty of this project by measuring multiple values.

Warm Up

- **Prompt:** Raise your hand if you've ever seen an ad on a website that seemed like it was targeted specifically at you. Maybe it was something for which you had recently searched, or a band that you already like.
- **Discuss:** How do websites know what ads to show to *you*?

Everything you do on the internet generates data about you—the sites you visit, ads you click on, even your location on Earth when you connect to the internet. This massive collection of data from computer users is known as *big data*, and by analyzing this big data, companies who don't actually know who you are can make a pretty good prediction about what you'll like, or what you won't like.

Scientists also use big data to run experiments on a massive scale to solve extremely large problems, such as decoding the human genome or searching for extraterrestrial life.

Activity 1: Modifying the App

Announce to the class that they're going to run an experiment today, but instead of everyone taking their own readings and analyzing only what they gathered, they're going to take a lead from the internet and gather big data (or maybe "medium data") by making a program that collects all of our data in one central location, which we can use later to analyze results.

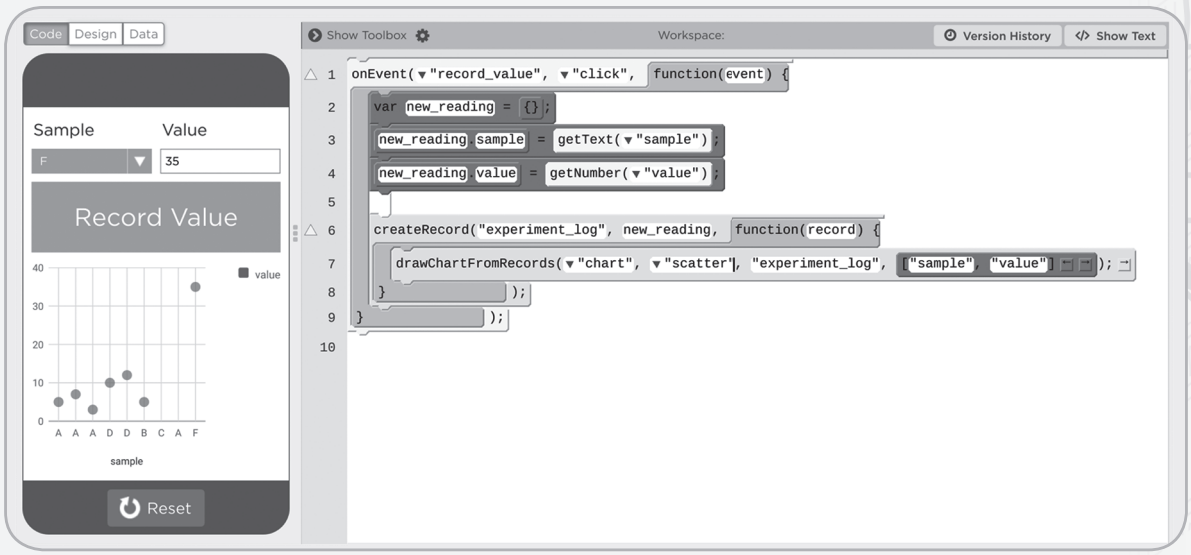

Figure 6.2: Lab Buddy starter code.

Remixing

Remixing is the act of taking an existing work, making your own copy, and then modifying the copy to meet your needs. In App Lab, you can create a remix by first viewing the code, and then clicking the Remix button.

Create a remix of the Lab Buddy starter project and share it with your students. Give them a chance to experiment with the program, entering whatever false

data they wish. As students explore the app, ask them to consider the following questions:

- What happens when you press the Record Value button?
- What does the chart at the bottom display?
- How could you change the chart so that it is more useful or meaningful?

Ask the class to share their thoughts about the preceding prompts. They should have noticed that when they clicked Record Value it added their data to the chart, but also that the chart contained data from other sources (the rest of the class). They likely found the chart meaningless as-is, but maybe adding a different type of visual, categorization, or filtering would make it more useful.

At this point, you can reveal the code that's running under the hood, and potentially start modifying it for your own needs. Have each study make a remix of your project by clicking the View Code and then Remix buttons. Give students a chance to read through the code on their own, and then walk through it as a class:

```
onEvent("record_value", "click", function(event) {
  var new_reading = {};
  new_reading.sample = getText("sample");
  new_reading.value = getNumber("value");

  createRecord("experiment_log", new_reading, function(record) {
    drawChartFromRecords("chart", "scatter", "experiment_log",
    ["sample", "value"]);
  });
});
```

- The main `onEvent()` block responds to the user clicking on the button with ID "record_value".
- The variable `new_reading` is an object that will let users collect multiple values in one variable that can be saved to our database.
- The property `new_reading.sample` adds the text in the dropdown with ID "sample" to the object `new_reading`.
- The property `new_reading.value` adds the number entered in the text field "value" to the object `new_reading`.