2018

Breaking Boundaries in Computing in Undergraduate Courses

Kam D. Dahlquist
Loyola Marymount University, kam.dahlquist@lmu.edu

John David N. Dionisio
Loyola Marymount University, dondi@lmu.edu

Ran Libeskind-Hadas

Anna E. Bargagliotti
Loyola Marymount University, Anna.Bargagliotti@lmu.edu

Journal of Research in STEM Education

Vol 4, No 1, July 2018, 81-100

RESEARCH REPORT

Breaking Boundaries in Computing in Undergraduate Courses

Kam D. Dahlquist[a], John David N. Dionisio[a], Ran Libeskind-Hadas[b], Anna Bargagliotti[a]

[a]*Loyola Marymount University, USA;* [b]*Harvey Mudd College, USA*

Abstract: *An important question in undergraduate curricula is that of incorporating computing into STEM courses for majors and non-majors alike. What does it mean to teach "computing" in this context? What are some of the benefits and challenges for students and instructors in such courses? This paper contributes to this important dialog by describing three undergraduate courses that have been developed and taught at Harvey Mudd College and Loyola Marymount University. Each case study describes the course objectives, implementation challenges, and assessments.*

Keywords: Computing, undergraduate courses, computational thinking, STEM courses

While computing has become an integral part of the STEM disciplines, best practices on how to incorporate computing effectively into a STEM classroom remains an open question. The distinction between the concept or paradigm—computing or computation—and the predominant tool for executing this paradigm—the computer—remains a point of debate or misunderstanding; there is only agreement that current problems in STEM fields benefit greatly from the integration of computation into the methods and thinking of these disciplines.

In many cases, computers are used as a tool to help with calculations and data collection or analysis. In other cases, computers are used to help demonstrate concepts. Still in other scenarios, computers are viewed as a vehicle to help discuss "computational thinking" as a problem-solving process. In general, many efforts have been made in recent years to teach computing concepts and skills in undergraduate STEM courses. Educators and researchers have engaged in a healthy debate on the benefits of such courses, and on the notion of what "computational thinking" is in the first place (Papert, 1980 and 1996; Wing, 2006; Barba et al., 2016). A key theme of the Breaking Boundaries conference was, therefore, to discuss some successful practices of integrating computing in undergraduate STEM courses and curricula.

This paper seeks to contribute to this ongoing dialog by using three case studies to both (1) explore the potential benefits and challenges to students in computing-infused STEM courses and (2) examine the potential benefits and challenges for faculty in designing and implementing such courses. Our hope is that the lessons that we have learned in these case studies will be useful to educators who are considering developing and implementing computationally rich STEM courses.

We begin by addressing the question: What does it mean to teach "computing" in a STEM course? As mentioned above, computation can mean many different things ranging from using computers to conducting virtual experiments to teaching programming. In our context, we mean teaching computational problem-solving skills that provide students with the ability and confidence to explore, analyze, and solve a range of problems that they have not seen before. This may involve using existing software tools, writing one's own programs from scratch, or developing algorithmic solutions. This is in contrast to using software to visualize

[a] Seaver College of Science and Engineering, Life Sciences Building 289, Los Angeles, CA, USA, E-mail: Kam.Dahlquist@lmu.edu

Dahlquist, K. D., Dionisio, J. D., Libeskind-Hadas, R., & Bargagliotti, A. (2018). Breaking Boundaries in Computing in Undergraduate Courses. *Journal of Research in STEM Education*, *4*(1), 81-100.

ISSN:2149-8504 (online)

the behavior of systems, to find solutions to a mathematical problem by running provided code, or varying parameters in existing software. While all such uses of computers are potentially valuable, the distinction that we draw in this paper is between using the computer as a tool to teach or convey a concept versus developing a set of broadly-applicable computational skills, techniques, and problem-solving strategies. In this context, we seek to better understand the benefits and challenges of such courses for both students and instructors.

We describe three case studies based on courses developed and taught at Harvey Mudd College (HMC) and Loyola Marymount University (LMU). These courses include a range of computational skills and techniques. The first case study describes a non-majors' introductory computer science course developed at HMC that covers foundational computing and programming concepts in the context of problems in biology. The second case study describes a non-majors statistics course developed at LMU that teaches concepts in statistics using student investigation of real datasets and a just-in-time delivery of material in response to questions posed by students. The third case study describes a biological databases course at LMU that brings upper-division computer science and biology majors together to explore research problems that involve expertise in both computing and biology.

For each case study, we begin by providing motivation for and background of the course, some details about the target audience and course logistics, and a discussion about the perceived benefits for students and the challenges for both students and the faculty teaching the course. Finally, we conclude with a synthesis of the lessons learned from these three case studies.

Case Study 1: Teaching Introductory Computer Science in a Biological Context

Ran Libeskind-Hadas

Department of Computer Science, Harvey Mudd College

Motivation and Background: Harvey Mudd College is a STEM-focused liberal arts college. As part of the college's common core curriculum, every student is required to take an introductory computer science course in the fall of their freshman year. We have developed a version of this course called "CS 5 Green" that uses applications from biology to motivate programming and computational problem-solving. This course covers nearly all of the computer science content in our "standard" introductory course but presents the material "just in time" to explore and solve problems that arise in the life sciences. The course is taught using the Python programming language both because that is the language used in our other introductory courses and because it provides a level of abstraction that permits writing interesting programs within the first few weeks of the course.

Learning Objectives: This course seeks to provide students with foundational and broadly-applicable programming and computational problem-solving skills. Our goal is to provide students with tools and techniques to formulate a computational problem precisely, design an algorithm to solve the problem, design a well-structured program to implement the algorithm, and learn to test and document the program.

Specifically, by the end of the course, students should be able to design, implement, and test programs that use control structures (if-then-else), loops (for and while), recursion, object-oriented design, and basic data structures. In addition, the course exposes students to foundational theoretical ideas including efficiency of algorithms and computability. Students should be able to explain these concepts and use them in the design of their own algorithm solutions when relevant.

All of the course content is motivated by questions and applications from biology, both to help bring the material to life and to demonstrate how computation is relevant to other STEM disciplines and to society in general.

Target Audience: The course is designed for students with no prior computing background and only a single high school biology course. Approximately 20 HMC first-year students (10% of the freshman class) elect to take this course. HMC students are invited to indicate their preference for one of four introductory courses that are offered (this course, the standard offering, or one of two more advanced sections for students with

ISSN:2149-8504 (online)

considerable prior computing experience). In addition, the course is available to approximately 20 declared life sciences majors (sophomores through seniors) from four other neighboring colleges (Pomona, Pitzer, Scripps, and Claremont McKenna) in the Claremont Colleges consortium.

Structure and Content of the Course: The course is structured as a sequence of 3-week modules, each of which presents a fundamental question in biology and then provides students with the programming concepts and tools that allow them to write programs (from scratch) that explore the motivating question. Ultimately, students use their programs to make their own discoveries on data that we provide.

The first module of the course opens with the story of Typhoid Mary and the discovery that typhoid is caused by the ingestion of Salmonella enterica typhi. Strangely, while this strain is pathogenic, other strains of salmonella are not. Why is this? With this question, we begin developing the programming concepts (basic data types, functions, loop structures) that allow students to write their own programs to explore the DNA sequences in bacterial chromosomes. Using their own programs, they search for differences in the pathogenic and non-pathogenic strains. In the second week of the course, students write their own programs to compare the GC content (the proportion of G and C nucleotides in the DNA) of strains of bacteria and discover that the pathogenic strain of salmonella has very different GC content from that of other bacteria. The following week, students write programs to identify the specific region of the salmonella genome where pathogenic and non-pathogenic strains differ and, ultimately, write programs that use randomization methods to identify the pathogenic genes themselves. Once the students discover these genes, they use the online BLAST tool to learn about the origins of these genes. Over the three-week duration of this module, students learn to write small programs that use basic data types, control flow (if-else), and loops (for and while loops).

The second module of the course poses the question "What determines the sex of an individual organism?" We explain that the sex determination systems in mammals and birds are different and ask if those systems evolved independently or had a common origin. Sequence alignment techniques provide a tool for answering this question. This module explains the concept of sequence alignment and introduces students to recursive sequence alignment algorithms. Students learn how the algorithms work, modify them to their needs, and implement them from scratch. Students quickly discover that these recursive algorithms are very slow, motivating concepts in efficient algorithm design. We show students the method of "memoization" (a simpler version of dynamic programming) and students use that method to accelerate their sequence alignment programs. Ultimately, students are able to use their programs to align avian and mammalian genes, identify orthologs, and use the algorithmic method of best reciprocal hits to answer the question of the origin of these two sex determination systems. In this module, student learn to write recursive functions, accelerate them using memoization, and learn and implement general-purpose algorithms (e.g., best reciprocal hits).

The third module explores the origins of modern humans using phylogenetics. Students learn a relatively simple phylogenetic inference algorithm (UPGMA) and use that algorithm to construct the phylogenetic trees for ancient and modern primates using provided data. Ultimately, students are able to find evidence for and against different theories of the evolution of humans and the relationship between modern humans and neanderthals. In this part of the course, students design, implement, and test larger programs that integrate the techniques that they have learned in the past modules.

In the final module, students learn foundations of object-oriented programming and use those concepts to write and conduct their own simulations of a variety of biological phenomena.

At the end of the course, students are presented with three choices for a 2-week final project that integrates many ideas from the course. For example, in one project, students use a maximum likelihood method to infer the regulatory network for a set of genes. These projects typically require approximately 200 lines of code and then ask students to use their programs to explore several provided datasets. Figure 1 shows two examples of student work.

ISSN:2149-8504 (online)

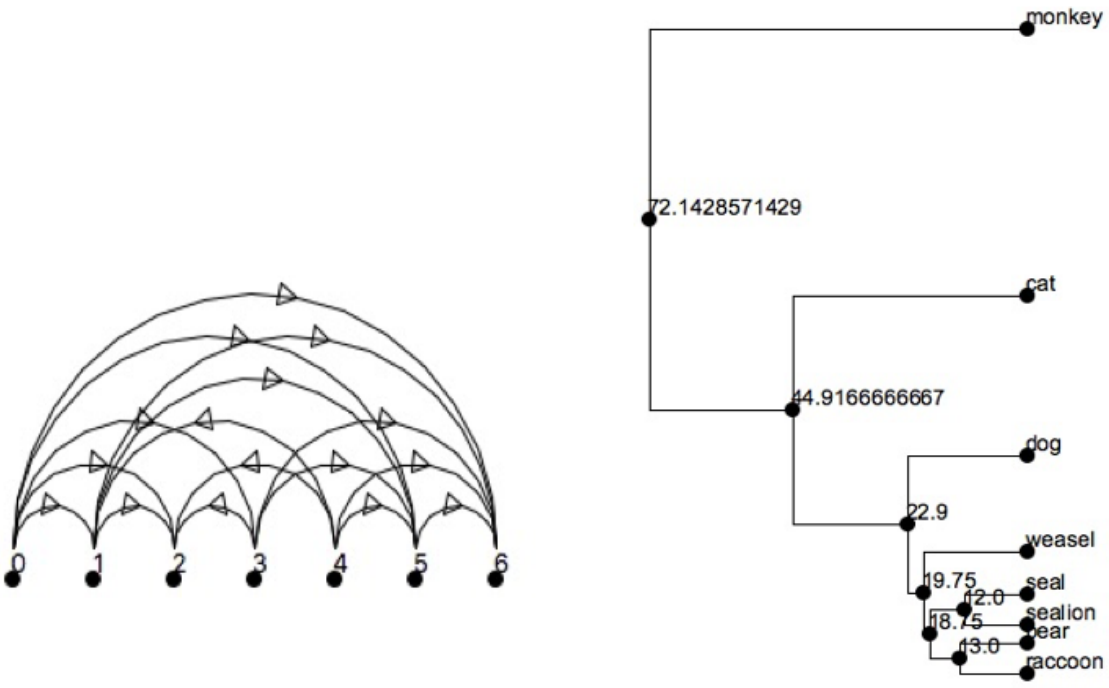


Figure 1. Examples of the output from student-written programs. (Left) A maximum likelihood gene regulatory network for six genes inferred from binary expression data. (Right) A phylogenetic tree inferred using the UPGMA algorithm.

Course Logistics: This is a one semester (14-week) course taught each fall. The course meets twice each week for 75 minutes per lecture and has an open 2-hour lab each Friday afternoon where students are encouraged to get started on their weekly assignments. Python's relatively simple syntax allows the instructor to use most of the class time to talk about principles.

The course has been taught since 2009 and is co-taught by a computer science professor and a biology professor. In the first several iterations of the course, the biology professor would give the first lecture each week to motivate the problem and introduce or foreshadow the basic computing approach to that problem. The second lecture would be given by the computer science professor and would dive into the computational tools required to solve the problems that week.

After several iterations of the course, the two instructors became sufficiently comfortable with all of the content. In recent offerings, the instructors have divided the material into multi-week units, with one instructor lecturing for two or three weeks of classes at a time. Both instructors are always present in class and in the Friday open labs. The course is now sufficiently mature that new instructors have begun teaching the course. In the near future, it is expected that one professor will teach this course alone (e.g., either a computer science professor or a biology professor).

Benefits and Challenges for Students: Students in this course gradually achieve proficiency with developing algorithmic solutions to computational problems and designing, implementing, and testing programs that implement those algorithmic solutions. Based on our assessment of their capstone student projects and performance on the final exam, students seem to develop a strong sense of confidence in transferring their computational competence to new domains. As one indication, the students were surveyed near the end of a recent offering of this course and asked them: "Have you used the skills learned in this class to write a program to solve a problem in a different class this semester when programming was not required for that assignment?" The overwhelming majority of students in the course responded in the affirmative.

Anecdotally, students who have chosen "CS 5 Green" have reported that their decision to take this course was based on (1) an interest in the life sciences, (2) trepidation about taking a more traditional computer science course, and (3) curiosity about the connections between computing and biology.

Life sciences students from the other Claremont Colleges have reported that (1) they recognize the importance of computing in the life sciences, (2) their academic advisor recommended this course, or (3) they are engaged in a research project (e.g, a senior thesis project) that requires some computational sophistication.

Our assessment is that the interdisciplinary nature of this course makes it more inviting for students who have misgivings about taking a traditional introductory computer science course. In addition, students with an interest in the life sciences are drawn to the course because of potential applications to their current or future work.

In end-of-semester course evaluations, students report working an average of 7.5/hours per week outside of class. However, 25% of students report working 10 or more hours per week outside of class. This is more than the "standard" introductory computer science course but still within the normal range for first-year courses at the college. Some students report that they perceive that the material in this course is "harder" than in the "standard" introductory course. Finally, while most students report that they find the connections between the computer science and the biology to be smooth and well-integrated, some students don't see those connections as clearly or find the integration of topics to be less smooth than desired.

Benefits and Challenges for the Instructors: One of the clear benefits of this course to the instructors is the joy of working with colleagues in other fields. The collaborations are both intellectually stimulating and have led to other interactions in both teaching and research.

There are also, of course, a number of challenges. We learned that for the first several iterations of the course, co-teaching did not result in half the work for each instructor. There was substantial overhead in planning the lectures, assignments, and exams. In addition, both professors were always in the classroom and lab together. Fortunately, both professors' home departments treated this course as a full course rather than a half course worth of teaching - at least for the first several years. This investment by the two departments was critical for the success of this course.

Another challenge is the variation in the backgrounds of the students. HMC first-year students have (at most) a year of high school biology background whereas the off-campus life sciences majors have typically completed several college-level biology courses. Conversely, HMC students are generally very comfortable with mathematical concepts (e.g., basic concepts in probability and mathematical logic) whereas that comfort-level among off-campus students is more varied. Most of these challenges were largely worked out by the second offering of the course, but this is an issue that requires constant vigilance on the part of the instructors. Instructors have learned to use motivating examples that are mostly unfamiliar even to the biology majors and to encourage students to work in pairs during class time exercises and some programming homework assignments. These strategies help "level the playing field" for the class and appear to mitigate most, but not all, of the disparities in student background.

Outcomes and Assessments: The course appears to be successful across a number of dimensions. For example, on the college's 7-point Likert scale teaching evaluations, the mean response to the question "This course stimulated my interest in the subject matter" was 6.69 (n = 26, college mean = 5.65). On the question "I learned a great deal in this course", the mean was 6.73 (n = 26, college mean= 5.92).

One outcome of this course is that there has been a growing interest in computational biology at the college. Several years after the first offering of this course, the college approved a new "Mathematical and Computational Biology" major. Many graduates from this major have gone to top PhD programs in computational biology and related areas while others have gone to work in life sciences companies.

ISSN:2149-8504 (online)

Case Study 2: Introductory Statistics through Investigation

Anna E. Bargagliotti

Department of Mathematics, Loyola Marymount University

Motivation and Background: LMU offers a general education introductory statistics class, Math 104, that fulfills a quantitative reasoning graduation requirement. The students in this course are non-STEM majors. Over the last five years, the course has been re-envisioned as an introductory statistics course for data-literate consumers. The thought process for this redesign is that students need exposure working with and understanding data that they will encounter in their daily lives. The course is thus organized as a project-based course in which each project brings to light different statistical topics and ideas students may encounter in their daily life. Each project, or "investigation", uses compelling and relevant data sets that engage students in learning statistics through current events.

Learning Objectives: The course seeks to provide students with the ability to carry out statistical investigations, to compute and carry out some statistical procedures (e.g., descriptive statistics, hypothesis testing, regression), and to discover how statistics fits into our lives. The course strives to demonstrate how statistics can be used to answer interesting and compelling questions with data.

Target Audience: The course is designed for non-STEM majors fulfilling a math general education graduation requirement. Typically, there are 25 students enrolled in each section. Student majors include dance, sociology, communications, and others.

Course Structure and Logistics: Two to three sections of Math 104 arentaught each semester (15 week). The sections of the course reported on in this paper meet twice each week for 75 minutes in a computer lab. The course is founded on three important, fundamental, and particularly timely themes in statistics. Students need to (1) employ technology, (2) explore real data sets, and (3) practice communicating statistical ideas and results. Moreover, statistics should be guided and taught through the statistical investigative process of formulating a question and collecting/considering appropriate data, choosing the appropriate analysis technique and interpreting the results to answer the question (Franklin et al, 2007). The material commonly taught in introductory statistics courses often focuses on techniques, but such methods are often "necessary but not sufficient" for modern data analysis (Hardin, Hoerl, Norton, & Nolan, 2015; Ridgeway, 2015). In contrast, this course is aimed towards the modern data consumer.

The course has a total of ten projects that are completed throughout the semester. The statistical ideas and techniques covered in a project are then used in subsequent projects. Each project is typically assigned on a Tuesday, lasts approximately one week (two class periods) and is submitted the following Tuesday. All work is done electronically –project assignments are given on a Powerpoint slide in the form of an investigative question and all projects are written up with a one- to two-page limit and submitted on DropBox. As an example, some of the project assignments from the fall of 2016 included the following investigative questions:

- Does texting while doing another activity distract from the task at hand?
- Do our presidential candidates say what we think they say? Are there differences in the way the presidential candidates Clinton and Trump speak?
- What are evidence-based suggestions to reduce the public landfill?
- What region or country of the world do you recommend providing aid to in order to reduce the world's child mortality rate?
- How does religion and region of the world impact the presence of terrorism?

For each question, students are expected to construct a data-supported write-up that follows the Guidelines for Assessment In Statistics Education (GAISE) investigative process of formulating a question,

collecting/considering data, analyzing data, and interpreting results (Franklin et al., 2007). The four components of the GAISE process are labeled in the student write-ups. Figure 2 illustrates a student's write-up that clearly shows the labeled four components of the GAISE report.

Each project is guided by an investigative question that is posed to the students as their only instruction. The goal of each project is to answer the investigative question in a coherent well-presented document. Students are given access to a relevant data set to help answer the investigative question at hand. This data set is either made available through a class group created in statistical software StatCrunch or may be accessed through a website. The course makes heavy use of StatCrunch, a point-and-click web-based statistical software package that is very easy to use.

Because the course is taught in a computer lab, students work on StatCrunch on a daily basis. The data include typically upward of 1000 observations and may or may not be cleaned. In addition, three of the ten investigations require students to interact with the data through apps (heat maps, multivariable scatterplots, etc.). Most of the data sets are multi-dimensional, offering many opportunities for multiple entry points for students to answer the investigative questions.

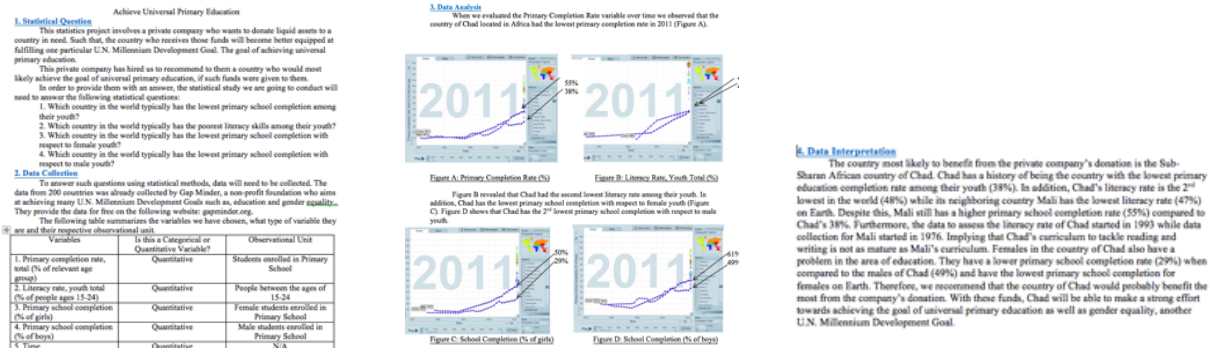


Figure 2. Example Student Write-Up with Labeled Components

All projects are group-based. Groups range from 2 to 4 people depending on the project. Each group is expected to turn in one final write-up for each project. The groups rotate for each project and thus students in the class typically work with every person in the class at least once during the semester. Group work is an appropriate pedagogy that aligns nicely with the course theme of communication.

Students work on their projects in class. Students are encouraged to ask questions in class and the instructor addresses the answer to the entire class. This "just in time" teaching provides students with relevant and necessary statistical content to help them complete the project. Often, these "teaching moments" comprise 10 to 20 minutes of lecture-style presentation where a question is expanded upon and several examples are given. Students are then asked to work through similar problems on their own before returning to their project.

Benefits and Challenges for Students: Initially students struggle to understand the expectations of the course. Most students choosing this course do so to fulfill their general education requirement for quantitative reasoning. Students enrolling in this course are typically "math-phobic" and many express anxiety and worry about being enrolled in a college math course. Because of the open-ended nature of the projects in the class, some students initially struggle to produce high quality work. Some students focus merely on the computation of statistics, insisting that they want formulas to compute. Some create graphical displays without reasoning about whether the graphical display is appropriate for their situation. Over the course of the term, however, students mature in both their statistical understanding and communication skills.

The structure of the class requires students to engage with the computer as an investigative tool to answer statistical questions. Consequently, students naturally engage in computational thinking through problem-solving. In addition, due to the emphasis on communication in the course, students view their computational skills as a toolkit for problem solving. Due to the open-ended nature of the projects, students develop creative

ISSN:2149-8504 (online)

problem-solving skills.

Benefits and Challenges for Instructors: The main challenge of teaching this course is the grading of the projects. Currently, no research-based rubric is available to grade open-ended statistics projects of the sort that are given in this class. There are two main dimensions that each project needs to be graded on: the correctness of the statistics and the clarity of the exposition. These two dimensions are often difficult to separate. For example, one may encounter a project that is well-written, convincing, and clear but contains some statistical errors.

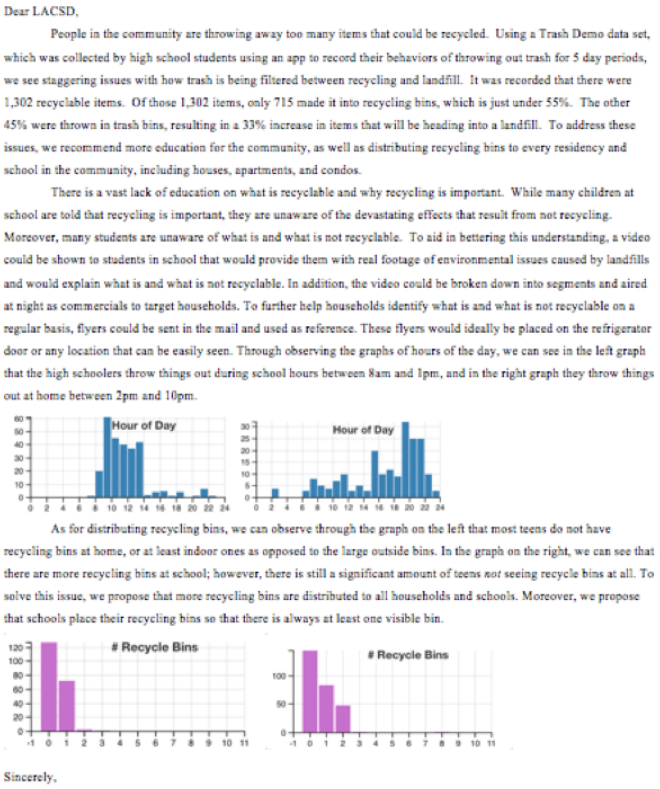
Recently, a new grading scheme was implemented in an effort to alleviate some of the difficulties and to provide important feedback to the students. A course grader was assigned to the course. For each project, the grader and the instructor separately graded each paper. The course grader read each paper for clarity of communication. The course grader was not provided with the assignment before grading and had to deduce the assignment from the student write-up. The clarity of the project was assigned a grade on a 5-point scale. Then, the instructor read the same submissions without seeing the grader's score. The instructor read the projects for correctness and assigned a grade on a 5-point scale. The two grades were added together and each assignment received a final grade (out of 10 possible points). The next iteration of this course will clarify the specific point assignments on each of the Likert scales. For example, a score of 5 on content means that no errors were committed in the write-up, a 4 might mean that there are 1-2 minor errors but no major ones, etc.

A second difficulty with grading is the printing of the projects. While one could opt to grade the projects electronically, it is difficult to navigate the logistics of grading electronically and then to return the graded projects to the students. With 25 students per section and ten projects of two pages each, the volume of printing is very large. While a notable effort was made to make the course "paperless," (the investigations were given on a Powerpoint slide and turned in electronically) printing of the projects was sometimes necessary. Students turned in assignments using Microsoft word, PDFs, and several other ways. Not all of these allow for track changes to record grades and comments. Therefore, at times, some assignments needed to be printed in order to be graded. This was slightly contradictory to the general philosophy of the course that was exposing students to statistics through technology. In the fall of 2016, approximately 20 total projects throughout the semester were printed.

Outcomes and Assessments: The course appears to be successful across a number of dimensions. Student attitudes are generally very positive towards statistics at the end of the course. For example, on the college's 5-point Likert scale teaching evaluations for the fall 2016 semester, the mean response to the question "This course stimulated my interest in the subject matter" was 4.03 in one section and 4.43 in another section ($n1 = 20$ & $n2=21$, department average= 3.49). Students were asked two open-ended questions: What did you find to be most beneficial about the course? And What would have made this course more effective for you? Across the two sections, the course received 25 positive comments and three negative comments. Of the positive comments, five of them specifically mentioned the projects. The other positive comments focused on the fact that the class connected statistics to the daily lives of the students and overall was effective and enjoyable. For example, students wrote "I learned how to apply statistics to my daily life" and "learning that stats IS applicable to everyday life," and yet another student mentioned " I found the hands-on learning to be beneficial." The negative comments were: "I felt like what was expected was unclear," "receiving [insufficient] feedback on my grades," and "the course would have been more effective if we were given less projects." These comments highlighted the difficulties noted above about the grading of the course. Providing constructive feedback for a course of this type is a challenge.

As the semester continued, student grades on the projects improved. While mini-project 1 had an average score of 6.2 (out of 10 points), the last project before the final raised the average to 7.7 points. Overall, the sophistication of student write-ups improved over time. For example, consider the work from group A who worked together on some of the projects throughout the semester. In mini-project 4 (Figure 3), the project where they wrote a letter to the LA county sanitation department, the group used simple histograms and bar

graphs and only referred to their graphics in a limited manner in the write-up. They only refer to the graphs minimally in their writing and also include other statistics that are not recoverable from the graphs.



Dear LACSD,

People in the community are throwing away too many items that could be recycled. Using a Trash Demo data set, which was collected by high school students using an app to record their behaviors of throwing out trash for 5 day periods, we see staggering issues with how trash is being filtered between recycling and landfill. It was recorded that there were 1,302 recyclable items. Of those 1,302 items, only 715 made it into recycling bins, which is just under 55%. The other 45% were thrown in trash bins, resulting in a 33% increase in items that will be heading into a landfill. To address these issues, we recommend more education for the community, as well as distributing recycling bins to every residency and school in the community, including houses, apartments, and condos.

There is a vast lack of education on what is recyclable and why recycling is important. While many children at school are told that recycling is important, they are unaware of the devastating effects that result from not recycling. Moreover, many students are unaware of what is and what is not recyclable. To aid in bettering this understanding, a video could be shown to students in school that would provide them with real footage of environmental issues caused by landfills and would explain what is and what is not recyclable. In addition, the video could be broken down into segments and aired at night as commercials to target households. To further help households identify what is and what is not recyclable on a regular basis, flyers could be sent in the mail and used as reference. These flyers would ideally be placed on the refrigerator door or any location that can be easily seen. Through observing the graphs of hours of the day, we can see in the left graph that the high schoolers throw things out during school hours between 8am and 1pm, and in the right graph they throw things out at home between 2pm and 10pm.

As for distributing recycling bins, we can observe through the graph on the left that most teens do not have recycling bins at home, or at least indoor ones as opposed to the large outside bins. In the graph on the right, we can see that there are more recycling bins at school; however, there is still a significant amount of teens *not* seeing recycle bins at all. To solve this issue, we propose that more recycling bins are distributed to all households and schools. Moreover, we propose that schools place their recycling bins so that there is always at least one visible bin.

Sincerely,

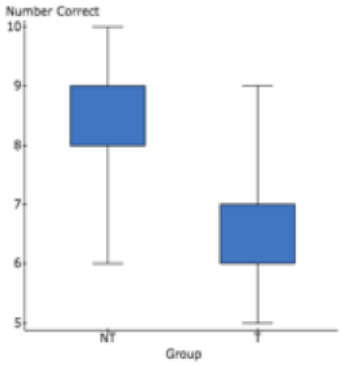
Figure 3. Group A mini-project 4

In the next mini-project (Figure 4), group A's analysis and interpretation integrated their graphics in a better way. For example, they discuss several aspects of the box-plot including overlap and IQR and utilize this information to draw their conclusion. As their work further progresses throughout the semester, their interpretations became more sophisticated as they notate more information in their output. In mini-project 8, they reference output and graphical displays and directly use it to support their answer. In this project, the group explored Redfin housing data they obtained from the internet to examine what factors impacted housing prices.

ISSN:2149-8504 (online)

Data Analysis

Looking specifically at the boxplot below, the Q1 for texting is 6 and the Q3 is 7, while for the non-texting group Q1 is 8 and the Q3 is 9.



Data Interpretation

Based on the data collected, it is evident that there is a significant difference between the scores of the non-texting and the texting groups. By looking at the boxplot, one can see that the spread of test scores for the non-texting group is higher than the spread of test scores for the texting group. This is supported by the fact that the Q1 for non-texting is 8 and the Q3 for texting is 7. In addition, the calculated difference between the average test scores is about 1.4 with the average test score for non-texting being 8.1 and the average test score for texting being 6.7. This difference is significant given that the chance of this randomly occurring is only 1% which was calculated in StatCrunch using the Z Stat function, in order to understand the possibility of this naturally occurring. In conclusion, texting significantly affects a person's ability to comprehend given information seeing as the scores between the non-texting and texting group do not overlap and the difference in means is extreme enough to be considered significant.

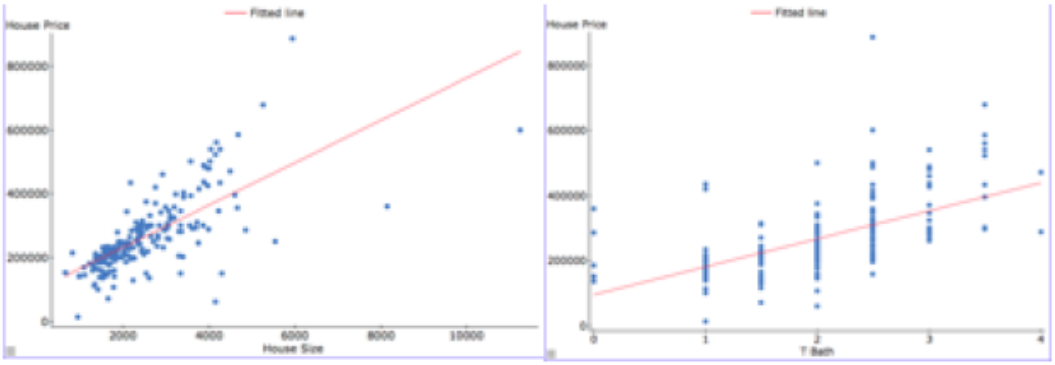
Figure 4. Group A Mini-Project 5

By using StatCrunch, we were able to determine the multiple linear regression and create two separate stat plots for house price and house size, and house price and number of bathrooms. The results are provided below.

Multiple linear regression results:
Dependent Variable: House Price
Independent Variable(s): House Size, T Bath
House Size = 29144.134 + 53.290598 House Size + 50569.315 T Bath

Parameter estimates:

Parameter	Estimate	Std. Err.	Alternative	DF	T-Stat	P-value
Intercept	29144.134	15643.365	≠ 0	197	1.8630348	0.0639
House Size	53.290598	4.6370349	≠ 0	197	11.492387	<0.0001
T Bath	50569.315	7414.0867	≠ 0	197	6.8207073	<0.0001



For this data set, the y-intercept is 29144.13. Both variables, house size, a quantitative variable, and number of bathrooms, a categorical variable, demonstrate a linear pattern; the relationship between both variables and house price is positive. The coefficient for house size is 53.29. The coefficient for number of bathrooms is 50569.32. The house size standard error is 4.64. The standard error for number of bathrooms is 7414.09. The confidence interval for house size is <.0001. The confidence interval for number of bathrooms is also <.0001.

On average, the base price of a house is $29,144.13. In looking at the multiple linear regression results, we can see that, on average, house price will increase by $53.29 for every additional square foot, given that the amount of bathrooms stays the same. We can also deduce that, on average, house price will increase by $50,569.32 for every additional bathroom, given that the house size stays the same. Given the P-value of <.0001 for house size, we can conclude that house size and house price are highly correlated. Given the P-value of <.0001 for number of bathrooms, we can also conclude that number of bathrooms and house price are also highly correlated. However, the standard error for bathrooms is very high in comparison to the error regarding house size. All of this information allows us to conclude that both bathrooms and house size have a significant relationship in regards to house size.

Figure 5. Group A Mini-Project 8

ISSN:2149-8504 (online)

Group A's progression, through mini-project 8 (Figure 5), serves as an example of what many groups achieved. As the course continued and their communication skills improved, mini-project write-ups became more complete. Because communication was one of the learning outcomes of the course, this improvement served as an indicator of overall success.

Case Study 3: A Biological Databases Course Promotes an Open Science Ecosystem for Teaching, Learning, and Research with Undergraduates

Kam D. Dahlquist

Department of Biology, Loyola Marymount University

John David N. Dionisio

Department of Electrical Engineering & Computer Science, Loyola Marymount University

Motivation and Background: The Biological Databases course that we have co-designed and co-taught at Loyola Marymount University (LMU) is a cross-listed upper-division biology and computer science course (BIOL/CMSI 367). It has evolved over the last nine years to promote an open science ecosystem (John R. Jungck, personal communication; Crouzier, 2015), encompassing both teaching and research with undergraduates (Figure 6). The initial course offering in Spring 2006 was a master's-level computer science course in bioinformatics that sprang from the co-authors' desire to initiate a teaching and research collaboration (i.e., implement the teacher-scholar model) around an open source bioinformatics project that would exemplify a pedagogy founded upon open source principles (Dionisio et al., 2007; Dionisio & Dahlquist, 2008). We shifted the focus of the course to undergraduates in Fall 2008 and have taught it six times over the last nine years, roughly on an every-other-year basis, due to departmental and college constraints on staffing team-taught courses.

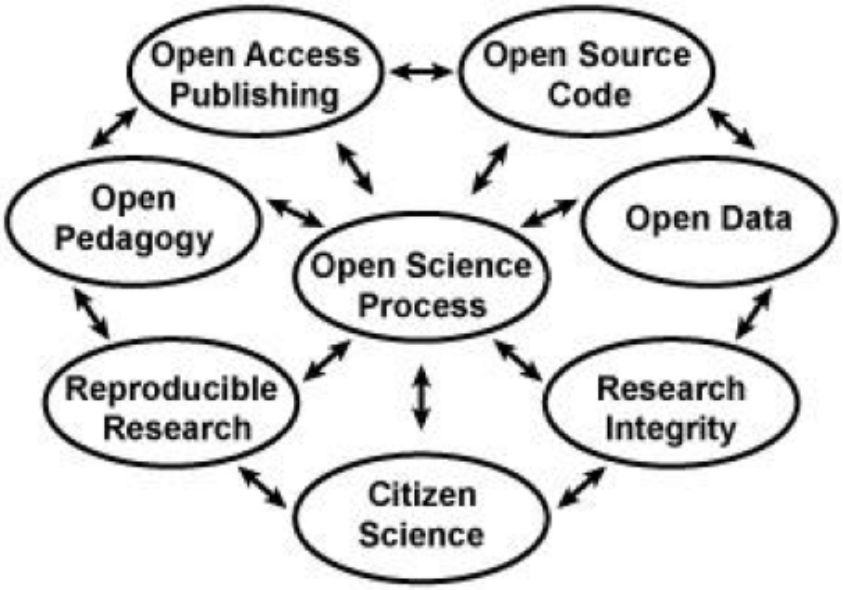


Figure 6: An open science ecosystem centered on an open science process of authentic research projects carried out in the Biological Databases course and in subsequent independent research with undergraduates. Open science relies upon the availability of open source code, open data, and open access to the scientific literature. Our public course website promotes an open pedagogy where students collaborate and learn science and engineering best practices with the benefit that resources are also available to other educators and the scientific community at large. Students learn how to conduct reproducible research and act with research integrity. They leave the course empowered to conduct citizen science, whether in graduate or professional school, or in other career paths. The products of student research in the course and research lab are then contributed back to the scientific community

as open source code, open data, and documentation in the form of reports and presentations, closing the loop on a meaningful experience that teaches them skills in information, data, and computer literacy. Arguably, this figure could be drawn as a fully-connected graph; we chose not to do this for the purposes of clarity.

The course culminates in a final project where the students are grouped in interdisciplinary teams of three or four to carry out an original research project. Their goal is to create a GenMAPP-compatible Gene Database (Dahlquist et al., 2002, Salomonis et al., 2007) using the open source XMLPipeDB software (Dionisio & Dahlquist, 2008) for a bacterial species not currently supported by GenMAPP, and then use the Gene Database to analyze published DNA microarray data for that species. For example, the Fall 2015 cohort created Gene Databases for Bordetella pertussis, Burkholderia cenocepacia, Shewanella oneidensis, and Shigella flexneri. Since 2008, students have created Gene Databases for 19 species. These student-generated Gene Databases are posted on our public code repository (first hosted by SourceForge, and now GitHub: https://github.com/lmu-bioinformatics/xmlpipedb) for use by the wider scientific community. This approach embodies the concept of a "problem space," which is the intersection of disciplinary knowledge, an open research question, and resources such as data and analysis tools to carry out a research project (Jungck et al., 2010). In support of this, course content is selected to directly serve whatever knowledge and skills will be needed by students to effectively execute the project. This is an example of "just-in-time" learning that has the advantage of being guided by expert "coaches" and utilizes the prior knowledge and skill set of students from complementary disciplines to form a learning community (Riel, 2000). Thus, biology topics include the fundamentals of genetics, molecular biology, and biochemistry needed to understand the data stored in biological databases, as well as the biotechnologies used to gather these data in a high-throughput manner. Computer science topics include what biological databases are, why they are important (and needed), the challenges that arise in compiling them effectively, and the relational database model.

Learning Objectives: The course's objectives are based upon L. Dee Fink's taxonomy of significant learning, which is divided into foundational knowledge, application, integration, the human dimension, caring, and learning how to learn (Fink, 2003). Long after the course concludes, it is hoped that students:

- Understand how biological information is encoded in the genome and can apply this knowledge to a variety of tasks and problems;
- Understand the core concepts, structure, and functions of a database, ranging from individual files to a full relational database management system, and can perform useful tasks with such data;
- Show discipline and proficiency in day-to-day science and engineering best practices, such as maintaining journals and notebooks, managing files and code, and critically evaluating scientific and technical information;
- Recognize and care about how the biological and technological issues presented in the course relate to and affect society, our daily lives, and ourselves;
- Have skills and tools for leaving one's comfort zone, flourishing outside of it, and learning more about biology and computer science on one's own; and
- Learn how to communicate and work effectively with colleagues from different disciplines.

Of special note are the last two objectives. At the beginning and throughout the semester, we emphasize to the students that leaving one's comfort zone, by definition, makes you uncomfortable, but that we are there to support them in the role of coaches, giving them guidance and opportunities to practice new skills, and cheering them on to achieve something that they did not know they could do. Taking on this role requires courage (Palmer, 1997) because in these courses we often find ourselves in the position of student, rather than expert, in the opposite discipline. Furthermore, the inevitable troubleshooting of technical and research issues can make the course appear as disorganized or "messy" instead of as running smoothly. We must be willing to show vulnerability to the students, trusting that such messiness and the resultant troubleshooting are actually

ISSN:2149-8504 (online)

creating teachable moments. We must accept that the students may not see the value of the hard work involved until well after the course has concluded.

The last objective is modeled by the co-taught nature of the course: it is hoped that students witness a firsthand example of this collaboration in the interactions we have with each other as a biologist and a computer scientist working toward common goals, including teaching the course effectively, assisting students with our complementary expertise, and mentoring the course's team research project, which occupies the last third of the semester. This objective is also supported by assigning homework partners from complementary disciplines each week.

Target Audience: Since LMU does not have an official degree or program in bioinformatics, the course is designed for upper-division biology and computer science majors, but has also been enrolled in by interested students in other majors (applied information management systems, chemistry, biochemistry, biomathematics, mathematics, mechanical engineering, psychology, and studio arts) as well as by students in the University Honors Program. Class size has ranged from 11 to its cap of 16, with the goal of producing teams of three or four to work on the research project for their designated bacterial species. Because at LMU instructors cannot assume that biology majors will have taken introductory computer science courses or that computer science majors will have taken introductory biology, there are no stated pre-requisites for the course except for upper-division standing in the Seaver College of Science and Engineering (a requirement we have waived in several cases). To attract students to the course, it is designed to fulfill multiple degree requirements. This course fulfills the three separate requirements in the LMU Core Curriculum as an "Integrations"-level Interdisciplinary Connections course carrying upper-division Information Literacy and Oral Communication "flags" (cf., http://academics.lmu.edu/media/lmuacademics/universitycorecurriculumfacultyresources/Core%20Document_RevisedFeb817.pdf). For biology majors, this course counts additionally as an upper-division molecular area requirement; for computer science majors, this course also counts as a science elective.

Course Structure and Logistics: The course meets twice a week for 75 minutes each session, for one semester (15 weeks). Office hours provide additional contact time as per usual and are designated as "technical support" time in case students have issues performing tasks or even just running the software required for the course. We note that the coursework involved could easily fill an additional 3- or 4-hour laboratory session. The addition of a lab session has not been sought, however, due to the disparate ways that the Biology and Computer Science programs assign teaching credit for laboratory courses.

The course is hosted on a MediaWiki site on an LMU computer science server. All assignments and student work are hosted on this wiki, which we made public with the Fall 2013 cohort. Materials from the most recently completed course in Fall 2015 can be viewed at: https://xmlpipedb.cs.lmu.edu/biodb/fall2015, while the current Fall 2017 course-in-progress (at the time of this writing) can be viewed at https://xmlpipedb.cs.lmu.edu/biodb/fall2017. Students complete weekly individual assignments and share reflections on the course wiki, which typically alternate between biology and computer science topics. The use of the course wiki for assignments is a gentle introduction to coding for non-computer science majors and serves as the first platform in the course from which to introduce the value of open source code and software engineering best practices. For example, the wiki demonstrates the concept of version control with the history of all contributions being readily available. We emphasize that students should work incrementally and annotate the summary field before saving a change to the wiki, by reporting back the number of contributions each student makes week-to-week, as well as the percentage of changes where the summary field was annotated. In addition, documentation of workflows via an electronic notebook kept on the wiki is part of every assignment. Collaborative work between the biology and computer science students is encouraged through assigning new homework partners each week to encourage students to take advantage of the domain-level expertise of the different majors. The shared reflections provide opportunities for metacognition about their individual learning processes, reflection on the interpersonal qualities needed for successful teamwork, and discussion of ethics case studies. Extensive feedback is provided by the instructors each week on each student's User Talk page. The actual grades are kept

private on LMU's learning management system (formerly Blackboard, now Brightspace). Working publicly in this manner demonstrates how an open science process works.

The course has three phases which roughly divide the 15 weeks evenly: 1) Building Blocks (Genetic Code and Manipulating Text); 2) Going Deeper (Gene Expression Data and Relational Databases); and 3) Integrating for Research (Gene Database Project). The first "Building Blocks" part of the course introduces foundational material in molecular biology and computer science. Instead of strict sequential content, however, the topics are chosen in a way that points specifically toward the research project at the end of the semester—i.e., they are chosen based on what students need to know in order to execute the research project effectively.

The "Going Deeper" second part of the course introduces new content and also contains a dry-run of the research project for a species with known results. Students build a GenMAPP-compatible Gene Database for Vibrio cholerae from the current raw XML sources from UniProt and Gene Ontology. They perform a statistical analysis of the results of a published DNA microarray study on pathogenic versus laboratory strains of the bacterium (Merrell et al., 2002). They then combine the statistical analysis results with the gene database in GenMAPP (Dahlquist et al., 2002; Salomonis et al., 2007) to determine whether they draw the same conclusions as the study's authors and to directly observe the effects of database changes over time on the analysis of the data. This exercise is a powerful example of the need for reproducible research because the exact analysis the original authors performed (Merrell et al., 1992) cannot be replicated due to a lack of information in the methods of that paper. Moreover, student groups use a Gene Database produced by the previous class' cohort and compare it to results they generated to see how changes in databases will change analyses over time.

To conclude the semester, in the third part of the course, "Integrating for Research", the students are grouped into teams of three or four which then perform the same gene database building and published study comparison exercise for a bacterial species that has yet to be represented in GenMAPP. The typical class size produces four such teams and thus four gene databases to study the results of four real-world bacterial studies. To carry out the project, each student on the team is assigned the role of coder, quality assurance officer (QA), or data analyst. Each team also chooses a project manager. Class meetings for the last month of the semester are then devoted to both team meetings and guild meetings to receive feedback and guidance on the project. Guilds are composed of students from different teams who play the same role on the project (inspired by Wright & Boggs, 2002). All coders and QA meet with Dr. Dionisio and all project managers and data analysts meet with Dr. Dahlquist. The students create team web sites on the course wiki to manage their projects and the deliverables: the Gene Database, documentation, and a processed DNA microarray dataset that they analyzed using their new database, a group report and presentation slides, and individual statements of work and reflections on learning.

The oral communication and information literacy assignments are integrated with the course content and final project. In the first information literacy assignment, groups of students review, evaluate, and present about an individual biological database chosen from among those reported in that year's annual Nucleic Acids Research Database Issue (e.g., Galperin, Fernández-Suárez, and Rigden, 2017) according to a provided rubric. In the second information literacy assignment, which was developed in collaboration with a librarian, the students must use online literature (e.g., NCBI PubMed, ISI Web of Science, Google Scholar) and gene expression databases (e.g., NCBI Gene Expression Omnibus, EBI ArrayExpress) to find the articles and data for their projects. This is followed by journal club presentations where the coder and QA officer present the genome sequencing paper for their species and the GenMAPP users present the microarray paper. The entire team delivers a final research presentation to the class.

In addition to these information literacy assignments, we also reflect on a case of scientific misconduct involving Dr. Anil Poti, formerly of Duke University, who manipulated microarray data to support his claim that he could determine which patients would respond better to a chemotherapeutic drug (Baggerly & Coombes, 2009; http://bioinformatics.mdanderson.org/Supplements/ReproRsch-All/Modified/StarterSet/;and citations within). This case comes at a crucial juncture in the semester where the students have just learned about the

ISSN:2149-8504 (online)

intricacies of analyzing microarray data and drives home the need for careful documentation in the interests of reproducible research and open science.

Challenges and Lessons Learned: Having taught this course roughly every other year since 2006, we have found the course to entail a large amount of work, for both the students and the instructors. It has already been noted in the first case study in this paper that co-teaching does not reduce the work involved in a course. We have been fortunate that this has been recognized by both our departments and the college, and we have each received a full unit of teaching credit each time we have taught the course. The fact that our course fulfills multiple university core curriculum and major requirements aids in justifying the team-teaching. The large workload falls into three main areas, grading of weekly assignments, dealing with technology issues, and troubleshooting the research projects themselves.

The stated learning objective of "showing discipline and proficiency in day-to-day science and engineering best practices, such as maintaining journals and notebooks, managing files and code, and critically evaluating scientific and technical information," will only be met if students are given opportunities for repeated practice and are given substantive feedback to motivate improvement and, eventually, self-regulation. With this in mind, the instructors provide detailed feedback on the weekly wiki assignments, assigning points to the software engineering best practices noted in the section on Course Structure and Logistics. Such detailed feedback is time-consuming, but necessary to hold students accountable for these "process" skills. Even so, we have found that only rarely does a student completely internalize what it means to keep a good electronic notebook; as instructors, we struggle with how to teach this skill. We surmise that students with a stronger sense of ownership of their learning process or the research project perform better in this area. We somewhat manage the workload by being particularly conscientious and timely graders at the beginning of the semester in the first phase of the course when expectations are being set and student habits are being formed. If we get behind on grading in the second and third phases of the course, we then deliver oral feedback to the class or individual students until we can catch up.

The second area that contributes to the workload for the instructors is the technology itself. Inevitably, the wiki server goes down a few hours before an assignment deadline each semester, sometimes for a few hours, once for a few weeks (where we had to move to a different hosting site). Monitoring email and communicating with students in timely fashion is key to alleviating student stress and frustration. Furthermore, since we use research-grade software, albeit open source, it is vital to have a computer lab with all software properly installed. We assist students with installing software on their own laptops, but have run into myriad issues with this over the years, which fall into the area of providing computer, instead of instructional, support. In recent years, we have been able to grant students card key access to the lab, which has alleviated some of the software installation issues. We note that while the current generation of students are considered "digital natives," modern operating systems have been so successful at easing system navigation such that many students have only vague notions of what files are and how they are stored in directories. The lack of a mental model for how computers function impedes some of the higher-level skills we are trying to teach, but is absolutely necessary to succeed at bioinformatics research. Indeed, "95% of bioinformatics is getting your data into the right file format" (Dahlquist et al., 2016).

The third contributor to the high workload for instructors and students in this course is the real-world research project that student teams carry out at the end of the course. The instructors choose the species for the research projects, but the students must find an appropriate published DNA microarray dataset to analyze and build the actual gene database. This type of real-world study is a double-edged sword. On one hand, it lends relevance and impact to the course content and the product of their research will be shared with the scientific community. On the other hand, the projects can vary widely in difficulty level, despite initial vetting of datasets by instructors. Particular challenges faced by student groups over the years include inadequate gene identifier mapping by the source databases and numerous issues with the microarray datasets, including missing samples, missing column headers, missing identifiers, and the inability to match samples reported in a published paper

to data posted to a published repository. Even without these unanticipated challenges, the data analysis pipeline for each project is unique, requiring customization of instructions for each team's coders, QA, and data analysts, especially in the areas of data cleaning and statistical analysis. The "realness" of the project demands genuine rigor and attention to detail, particularly with taking notes and recording processes and results—a skill which as we note above is genuinely underpracticed by some students. In the end, the projects become an object lesson in what (not) to do when sharing open data, reinforcing why best practices are needed, and relating back to materials we use from DataONE.org earlier in the course (https://www.dataone.org/education-modules).

A final challenge with conducting this course "in the open," where all student work is visible to everyone in the course, is dealing with cases of academic dishonesty. We have had cases where students have directly copied work from other students and turned it in as their own. While the ease of copying and pasting from the wiki enabled the plagiarism in the first place, the date/time-stamping of each change in the wiki's history also enables determining the direction of copying. Each time we have taught the course, we have sought to improve the clarity of what is allowed and not allowed. In the current semester, we discussed some academic honesty case studies up front. In addition, we now require an "Acknowledgments" and "References" section for every student-generated wiki page. Students must acknowledge those with whom they have worked, and any website from which they borrowed "syntax" (wiki style or coding). They must include an academic honesty statement (While I worked with the people noted above, this individual journal entry was completed by me and not copied from another source) and sign it with their wiki signature. The References section must include a properly formatted citation for any source from which they borrowed content. Furthermore, by connecting their own work to the case of scientific fraud we examine in the course, we hope to instill in them the values of the scientific community and to demonstrate how dishonesty can do harm.

Outcomes and Assessments: In addition to rubrics for course deliverables, we employ additional strategies focused on making group assessment as fair and thorough as possible. Presentations are evaluated by student peers and not just the instructors: students are provided with half-page questionnaires for every presentation not made by them, with questions framed to encourage constructive criticism and reflection. Deliverables for the final project include an explicit "Statement of Work" item where each student expresses in writing the specific contributions that they made toward the project.

The assessment of learning outcomes would be greatly aided by a pre- and post-course survey instrument that could measure learning gains in the areas of computer, data, and information literacy. To our knowledge, such an instrument does not exist. The Course Undergraduate Research Experience (CURE) survey is geared towards more traditional laboratory work in biology (Denofrio et al., 2007), and the Research on the Integrated Science Curriculum (RISC) survey also does not seem to capture what we do in this course (http://www.grinnell.edu/academics/areas/psychology/assessments/risc-survey). We also feel that student course evaluations are a limited way to measure learning outcomes, especially for a course such as this. Over the years, the evaluations have been influenced by factors such as technical hurdles faced, unanticipated difficulties of the final project, class dynamics, students' intrinsic interest in the course at the outset, and whether the evaluations were combined or separated by instructor. In recent years, we have improved our recruitment efforts to find interested students and we have also learned to manage student expectations. We consistently remind them that they should expect some discomfort from "leaving their comfort zone" and that conducting real-world research projects is difficult, yet rewarding. As a result, for the 2015 course evaluations, all of the numerical scores for every item beat the Biology Department-, College-, and University-wide averages. The lowest score of 4.53 (out of 5) was for "increased interest in subject matter" and the highest score of 4.93 was for "instructor available for discussion." The overall effectiveness of instruction was 4.60, which was substantially improved from the 3.93 of Fall 2013. The written comments remarked upon the challenging pace of work with assignments due every week and gave some constructive criticism as to the structuring of due dates. These critiques were also paired with praise for some of the course design principles of homework partners and collaborative work, however. The 2015 evaluations matched what we both felt as instructors, that the course really "clicked" that year and that students achieved far more than any previous class. We are wary, though, of making the course

ISSN:2149-8504 (online)

too "frictionless," i.e., making it run so smoothly that few hurdles need be overcome by students. We believe that "messiness" has value. Indeed, we have had to retire the long-standing XMLPipeDB project because the GenMAPP software (Dahlquist et al., 2002; Salomonis et al., 2007) on which it was based was finally rendered non-operational by a Windows update sometime in early 2017. Instead, we are designing a new research project for the students centered on our open source GRNsight software (Dahlquist et al., 2016).

One important outcome of the course: it has provided the opportunity for many students, who might not have been involved in research during their undergraduate careers, to conduct an authentic, interdisciplinary research project. Of those, at least a dozen continued the research projects begun in the course as independent study in subsequent semesters, leading to conference presentations and capstone projects for both the biology and computer science majors. One of these students complained early in the semester about the difficulty of the work, but by the end turned around, feeling amazed and proud of what he accomplished, and joined our research team. It is these student outcomes that inspire us to do this work.

Conclusion

We hope that continuing innovation in undergraduate STEM education will continue breaking boundaries: boundaries that inhibit student entry into STEM fields due to negative preconceptions; boundaries that artificially separate STEM disciplines; and boundaries that limit creative exploration due to a lack of powerful computational and statistical toolkits. As computers become increasingly important in the STEM disciplines, it is important to investigate optimal uses of computers in the classroom. Beginning with Papert's seminal work and Wing's widely-referenced 2006 follow-up, the call for computational thinking as an important skill for the 21st century has resonated with many. This paper highlights three case studies that view computational thinking as a problem-solving process, beyond merely learning how to use or program a computer. All three courses utilized and developed computational thinking in their students through the creative problem-solving process--students were given a broad range of problems and tasks for which they had to employ the computer to help them solve.

The three case studies described in this paper include an introductory computing course for STEM majors, a "hands on" statistics course for students from all majors, and an upper-division course that brings computer science and biology majors together to explore open-ended research problems. While the levels, audiences, and objectives are different, these courses share an important common feature: challenging concepts are brought to life via computing through the use of compelling applications and creative explorations using real data. The case studies demonstrate that while bringing computing into courses in biology and statistics may seem interdisciplinary at first, such computing can actually be an integral part of the learning of disciplinary concepts. We hope that these case studies help point a way forward towards the integration of computing within the STEM disciplines.

Acknowledgments

Ran Libeskind-Hadas acknowledges generous support from HHMI for the development of the CS "Green" course at Harvey Mudd College. Anna Bargagliotti would like to thank the LMU Mathematics Department for its flexibility in reenvisioning the Math 104 course. Kam D. Dahlquist and John David N. Dionisio would like to thank our respective departments and the Frank R. Seaver College of Science and Engineering for affording us the repeated opportunity to team-teach the Biological Databases course. We thank Glenn Johnson-Grau of the William H. Hannon Library at Loyola Marymount University for guest lecturing and advising us on the information literacy assignments in the course. Finally, we would like to thank John R. Jungck for his guidance and support. The authors thank the anonymous reviewers whose comments and suggestions were invaluable in improving this paper.

References

Baggerly, K. A., & Coombes, K. R. (2009). Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology. *The Annals of Applied Statistics*, 1309-1334.

Baggerly, K. A., & Coombes, K. R. (n.d.) *"Starter Set" Materials for the saga*. Retrieved from http://bioinformatics.mdanderson.org/Supplements/ReproRsch-All/Modified/StarterSet/

Barba, L. et al. (2016). "*Computational thinking: I do not think it means what you think it means.*" Retrieved from http://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/

Crouzier, T. (2015). Science ecosystem 2.0: how will change occur. European Commission Directorate-General for Research and Innovation Report.

Dahlquist, K. D., Aikens, M. L., Dauer, J. T., Donovan, S. S., Eaton, C. D., Highlander, H. C., Jenkins, K. P., Jungck, J. R., LaMar, M. D., Ledder, G., Mayes, R. L., Schugart, R. C. (2017) *An invitation to modeling: Building a community with shared explicit practices*. PeerJ Preprints

Dahlquist, K. D. & Dionisio, J. D. N. (2015) Loyola Marymount University BIOL/CMSI 367-01: Biological databases Fall 2015. Retrieved from https://xmlpipedb.cs.lmu.edu/biodb/fall2015/index.php/Main_Page

Dahlquist, K. D. & Dionisio, J. D. N. (2017) Loyola Marymount University BIOL/CMSI 367-01: Biological databases Fall 2017. Retrieved from https://xmlpipedb.cs.lmu.edu/biodb/fall2017/index.php/Main_Page

Dahlquist, K. D., Dionisio, J. D. N., Fitzpatrick, B. G., Anguiano, N. A., Varshneya, A., Southwick, B. J., & Samdarshi, M. (2016) GRNsight: a web application and service for visualizing models of small- to medium-scale gene regulatory networks. PeerJ Computer Science 2:e85.

Dahlquist, K. D., Salomonis, N., Vranizan, K., Lawlor, S. C., & Conklin, B. R. (2002). GenMAPP, a new tool for viewing and analyzing microarray data on biological pathways. *Nature Genetics*, *31*(1), 19-20.

DataONE (2016) Education Modules. Retrieved from https://www.dataone.org/education-modules

Denofrio, L.A., Russell, B., Lopatto, D., & Lu, Y. (2007). Linking student interests to science curricula. Science, 318, 1872-1873. doi: 10.1126/science.1150788

Dionisio, J. D. N., & Dahlquist, K. D. (2008). Improving the computer science in bioinformatics through open source pedagogy. ACM SIGCSE Bulletin, 40(2), 115-119. doi: 10.1145/1383602.1383648

Dionisio, J. D. N., Dickson, C. L., August, S. E., Dorin, P. M., & Toal, R. (2007). An open source software culture in the undergraduate computer science curriculum. *ACM SIGCSE Bulletin*, *39*(2), 70-74.

Eaton, C. D., Callendar, H. L., Dahlquist, K.D., LaMar, M. D., Ledder, G., Schugart, R. C. (2017). A "Rule of Five" Framework for models and modeling to unify mathematicians and biologists and improve student learning, https://arxiv.org/abs/1607.02165v2.

Fink, L. D. (2003). *Creating significant learning experiences: An integrated approach to designing college courses*. San Francisco: John Wiley & Sons.

Franklin, C., Kader, G., Mewborn, D., Moreno, J., Peck, R., Perry, M., & Scheaffer, R. (2007). Guidelines for assessment and instruction in statistics education (GAISE) report: A pre-K–12 curriculum framework. Alexandria, VA: American Statistical Association. (Also available at www.amstat.org.)

Galperin, M. Y., Fernández-Suárez, X. M., & Rigden, D. J. (2017). The 24th annual nucleic acids research database issue: a look back and upcoming changes. *Nucleic Acids Research*, *45*(D1), D1-D11.

Hardin, J., Hoerl, R., Norton, N., & Nolan, D. (2015). Data science in statistics curricula: Preparing students to "Think with data," working paper, Retrieved from http://arxiv.org/ftp/arxiv/papers/1410/1410.3127.pdf

Jungck, J. R., Donovan, S. S., Weisstein, A. E., Khiripet, N., & Everse, S. J. (2010). Bioinformatics education dissemination with an evolutionary problem-solving perspective. *Briefings in Bioinformatics*, *11*(6), 570-581.

ISSN:2149-8504 (online)

lmu-bioinformatics (2016) xmlpipedb repository. Retrieved from https://github.com/lmu-bioinformatics/xmlpipedb

Lopatto, D. (2013) RISC Survey. Retrieved from http://www.grinnell.edu/academics/areas/psychology/assessments/risc-survey

Loyola Marymount University. (2011). New university core curriculum. Retrieved from http://academics.lmu.edu/media/lmuacademics/universitycorecurriculumfacultyresources/Core%20Document_RevisedFeb817.pdf

Merrell, D. S., Butler, S. M., Qadri, F., Dolganov, N. A., Alam, A., Cohen, M. B., ... & Camilli, A. (2002). Host-induced epidemic spread of the cholera bacterium. *Nature*, *417*(6889), 642.

Palmer, P. (1997). *The Courage to Teach*, 1st edition, Jossey-Bass ISBN 978-0787910587

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Basic Books, Inc., New York, NY.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95-123.

Riel, M. (2000). Education in the 21st century: Just-in-time learning or learning communities. In Fourth Annual Conference of the Emirates Center for Strategic Studies and Research, Abu Dhabi.

Ridgeway, J. (2015). Implications of the data revolution for statistics education. *International Statistical Review*, doi:10.1111/insr.12110

Salomonis, N., Hanspers, K., Zambon, A. C., Vranizan, K., Lawlor, S. C., Dahlquist, K. D., ... & Pico, A. R. (2007). GenMAPP 2: new features and resources for pathway analysis. *BMC Bioinformatics*, *8*(1), 217.

Wright, R., & Boggs, J. (2002). Learning cell biology as a team: a project-based approach to upper-division cell biology. *Cell Biology Education*, *1*(4), 145-S27.

Wing, J. (2006). "Computational thinking." *Communications of the ACM*, *49*(3), 33–35.