Introductory Programming Meets the Real World: Using Real Problems and Data in CS1

Ruth Anderson[1] Michael D. Ernst[1] Robert Ordóñez[2] Paul Pham[3] Steven A. Wolfman[4]
[1]University of Washington [2]Southern Adventist University [3]The Evergreen State College [4]U. of British Columbia
{rea,mernst}@cs.uw.edu rordonez@southern.edu phamp@evergreen.edu wolf@cs.ubc.ca

1. Summary

Too many students in introductory programming classes fail to understand the significance and utility of the concepts being taught. Their low motivation impacts their learning. One contributing factor is pedagogy that emphasizes computing for its own sake and assignments that are abstract, such as computing the factorial function.

Many educators have improved on such traditional approaches by teaching concepts in contexts that students find more relevant, such as games, robots, and media. Now, it is time to take the next step.

In this special session, participants will develop and discuss ways to teach introductory programming by means of real-world data analysis problems from science, engineering, business, and the humanities. Students can be motivated to learn programming in order to analyze DNA, predict the outcome of elections, detect fraudulent data, suggest friends in a social network, determine the authorship of texts, and more (see Section 3.4 for more examples). The approach is more than just a collection of "nifty assignments": rather, it affects the choice of topics and pedagogy, all of which together lead to greater student satisfaction.

The approach has been successfully used at 4 colleges and universities. The classes were effective for both CS and non-CS majors. Neither the computing material nor the problems need to be "dumbed down". At the end of the term students were amazed and delighted at the real data analysis that they could perform. They were excited about applying computation in their work and about learning more.

The special session contains a mix of activities, including comparative analysis of introductory classes; group discussion of curriculum design; a mini-panel discussing how the approach has worked in practice; and brainstorming about example assignments and curriculum revision.

Categories and Subject Descriptors: K.3.2 [**Computers and Education**]: Computer and Information Science Education—*Computer science education*; J.0 [**Computer Applications**]: General

General Terms: design, documentation, languages, measurement

Keywords: CS1, introductory computing, data programming, data processing

2. Audience and objective

This session is intended for instructors who teach introductory computing (CS1) to CS majors, non-CS majors, or a mix. It is also relevant to those responsible for curriculum design.

In this SIGCSE Special Session, participants will learn:

- The benefits of an approach grounded in real-world problems and data.
- How the real-world approach is more than a collection of "nifty assignments"; rather, it affects the choice of topics and pedagogy.
- The challenges and successes of applying this approach at multiple colleges, including pitfalls to avoid.
- Principles of creating an assignment or adapting an existing one to support and reinforce the computing concepts.

Participants will be given a packet of materials to take home, as well as online links to web resources including assignments, syllabi, handouts, and more.

Participants will leave the session prepared to make positive curricular change at their own institutions, whether creating a new class, adapting one from another institution to your curriculum, or revising your current offerings.

SIGCSE'14, March 5–8, 2014, Atlanta, Georgia, USA.
ACM 978-1-4503-2605-6/14/03.
http://dx.doi.org/10.1145/2538862.2538994.

3. Outline

A brief outline of the session follows.

- Description of, and motivation for, the overall approach including example end-of-term student projects (15 mins)
- Group discussion of pros and cons of this and other approaches to CS1 including appropriateness for different students, place in the curriculum, etc. (15 mins)
- Project idea brainstorming. (20 mins)
- Experience reports and Q and A. Instructors who have applied the approach will report on its successes and failures and will offer advice to those who wish to adopt it. (20 mins)
- Creation of support groups for followup after SIGCSE is over, to help instructors put their ideas into practice. Also, all discussions and ideas during the session will be collected, summarized, and shared on a publicly-accessible wiki. (5 mins)

3.1 A real-world approach to CS education

The key idea of our approach is that *every assignment uses an existing real-world dataset to answer a question that is relevant to science, engineering, or business*. Neither the questions nor the datasets are artificial.

This approach serves the following goals:

- motivate and excite students about computation
- enable students to process data to solve real problems
- teach computer programming
- teach computer science concepts
- serve both CS and non-CS majors

3.2 Comparison to other teaching approaches

Traditional introductory programming classes take their examples and assignments from the domains of puzzles, games, and abstract mathematics. For instance, students might be shown how to reverse a list or assigned to compute the Fibonacci sequence. This puzzles-and-games approach works well to motivate some students, but it

alienates others. Students find it difficult to appreciate the practical importance of the skills they are learning.

Recognizing the deficiencies of the puzzles-and-games approach, other classes have incorporated realistic demonstrations and assignments. We acknowledge our debt to these pioneering classes, which provide inspiration and examples. At the same time, we aspire to take the next step.

Media Computation [7, 9, 13] introduces students to images, sounds, and HTML pages, then shows students how to manipulate these data. The problem domain is more connected to entertainment and computers than to real-world problems. Other classes share this theme. Stevenson's real-world programming assignments [10, 11] are a web crawler, spam evaluator, and steganography. DePasquale [2] presents three data sources: stock quotes, and the APIs from Google and Slashdot.

The ITiCSE working group [4] offers 14 projects related to "social good" that motivate students and provide them with skills to solve complex problems; Erkan et al. [3] do the same for a data structures class. We weave the themes throughout the class. Similarly, Isbell et al. [5] combine mathematics, computation, and experiments to solve engineering problems.

Some classes that address realistic problems and data focus on visualization [8], statistics [1], or databases [12]. By contrast, we focus on computation and programming.

3.3 Effect on pedagogy

In order to support students' experience with realistic datasets, it is desirable to make a few changes to the topics, order, and presentation of traditional introductory material. Here are a few examples.

We introduce the foreach loop but not the for loop with an explicit index. Students use data structures, but they do not re-implement basic data structures [6]. We use GUIs solely for plotting, but never for pictures or animation. We do not create GUIs nor any other user interface. We use file I/O but not console I/O. We introduce recursion at the end of the term, as an enrichment topic, but it is not used in the assignments.

We use the Python language, because it is easy to use and is widely adopted in the sciences. Because Python has a significant and usable procedural subset, we do not discuss object orientation. We discuss the notion of a type extensively, but not static typing nor compilation.

We introduce a few concepts that are often missing from CS1 classes but are desirable for data analysis, such as basic statistics and how to plot a graph.

Overall, students learn the same concepts as in any other CS1 class, but a slightly different toolset that enables them to do better-motivated tasks. Our choice of topics is motivated by viewing our class not just as CS1 but also as CSΩ: it should give a solid foundation for subsequent practical and theoretical work, but should also be useful even if the student never takes another computer science class.

3.4 Example assignments and projects

The presenters and audience will discuss some of the assignments that we have used and will give away handouts for others. The assignments illustrate the class philosophy of using real datasets and problems. This runs deeper than using some nifty assignment, but the assignments could be repurposed for an existing CS class.

Given files of DNA data, categorize organisms according to their GC count. Along the way, the students discover the need for data cleaning and perform that step.

Determine which environmental factors affect Archaea population biodiversity. Archaea are single-celled microorganisms, and the environmental factors include ocean salinity, temperatures, and chemical concentrations. Students learn to process tabular data.

Given polling data leading up to U.S. presidential elections, replicate the election prediction performed by Nate Silver of the FiveThirtyEight blog. This requires ranking, weighting, and averaging, and it illustrates the difference between sampling error and pollster bias.

Given Facebook social network data, create multiple algorithms for recommending friends. Evaluating them on the real data via Monte Carlo methods shows the utility of randomization.

Statistically analyze the least significant digits of election results from the U.S. and Iran to detect fraud that is revealed as anomalies against the expected uniform distribution.

Statistically analyze the most significant digits geographical data from the real world and from literary fiction to detect anomalies from the expected Benford's-Law distribution. The two statistics assignments illustrate statistical methodology via a modeling and sampling approach, without a long detour into mathematics. It also shows the importance of considering sample size when determining whether a given sample is anomalous. They also show how abstraction permits problems that seem different to share most of their code.

Linguistically analyze texts to determine their authenticity: clean noisy transcriptions and compare the vocabulary and phrases with known-authentic texts. This shows how to use data structures and control flow constructs to write concise, powerful, understandable code.

Analyze the mood, or sentiment, of Twitter posts to determine correlations with geography. Other known correlations include with biological rhythms and economic indicators.

In some of our offerings, we have given students the opportunity to define a scientific question about a real-world dataset of their own. The final projects from one offering (`http://courses.cs.washington.edu/courses/cse140/13wi/projects/`) include:

· Examining decay mechanisms of the Higgs Boson
· What makes H5N1 avian influenza "Avian"? (Why not "H5N1 tiger influenza"?)
· Spatial variability in the Antarctic circumpolar current in a global climate model
· Evaluating the effectiveness of the European Union Emissions Trading System
· Local causes, public amenities, and poverty level
· Urban growth and unemployment
· Correlation between statistics and wins in major league baseball
· Payroll and success in Major League Baseball
· An iterative strength-based model for the prediction of NCAA basketball games

References

[1] R. Catrambone and M. Guzdial. Computational freakonomics. `http://swiki.cc.gatech.edu/compfreak`, 2012.

[2] P. DePasquale. Exploiting on-line data sources as the basis of programming projects. In *SIGCSE*, pages 283–287, 2006.

[3] A. Erkan, T. Pfaff, J. Hamilton, and M. Rogers. Sustainability themed problem solving in data structures and algorithms. In *SIGCSE*, pages 9–14, 2012.

[4] M. Goldweber, J. Barr, T. Clear, R. Davoli, S. Mann, E. Patitsas, and S. Portnoff. A framework for enhancing the social good in computing education: a values approach. *ACM Inroads*, 4(1):58–79, Mar. 2013.

[5] C. L. Isbell, L. A. Stein, R. Cutler, J. Forbes, L. Fraser, J. Impagliazzo, V. Proulx, S. Russ, R. Thomas, and Y. Xu. (Re)defining computing curricula by (re)defining computing. *SIGCSE Bull.*, 41(4):195–207, Jan. 2010.

[6] H. P. Langtangen. *A Primer on Scientific Programming with Python*. Springer, 3rd edition, 2012.

[7] L. Rich, H. Perry, and M. Guzdial. A CS1 course designed to address interests of women. In *SIGCSE*, pages 190–194, 2004.

[8] K. A. Robbins, D. M. Senseman, and P. E. Pate. Teaching biologists to compute using data visualization. In *SIGCSE*, pages 335–340, 2011.

[9] B. Simon, P. Kinnunen, L. Porter, and D. Zazkis. Experience report: CS1 for majors with media computation. In *ITiCSE*, pages 214–218, 2010.

[10] D. E. Stevenson and P. J. Wagner. Developing real-world programming assignments for CS1. In *ITICSE*, pages 158–162, 2006.

[11] D. E. Stevenson, M. R. Wick, and S. J. Ratering. Steganography and cartography: interesting assignments that reinforce machine representation, bit manipulation, and discrete structures concepts. *SIGCSE Bull.*, 37(1):277–281, Feb. 2005.

[12] D. G. Sullivan. A data-centric introduction to computer science for non-majors. In *SIGCSE*, pages 71–76, 2013.

[13] S. L. Tanimoto. *An Interdisciplinary Introduction to Image Processing: Pixels, Numbers, and Programs*. MIT Press, 2012.