Data Mining a Computational Thinking Data Science Course

Austin Cory Bart, Javier Tibau,
Eli Tilevich, Dennis Kafura, Clifford A. Shaffer

May 15, 2016

Abstract

As students of all majors and career paths are now expected to learn about computing, computer science education is challenged by the unprecedented scale and complexity of the effort required to address this requirement. Many institutions now offer courses in "Computational Thinking", which cover foundational computing concepts, including algorithm development, data abstraction, and computing ethics to educate diverse majors from the sciences, arts, humanities, and other fields. Although the material covered may not be as rigorous or in-depth as traditional introductory computing courses, thinking computationally is new for these students and they may suffer from low self-efficacy about their ability. Further, as non-major students often fail to immediately appreciate how computing can benefit their long-term career, their resulting motivation level often proves insufficient to see them through the more challenging aspects of the curriculum. Hence, courses that serve these students need to focus on both cognitive and motivational concerns, using new techniques to identify which students are struggling as early as possible, so that interventions can be staged to rescue and guide learners.

Our approach to the Computational Thinking course focuses on Abstraction (representing real-world objects with quantifiable properties), Algorithms (manipulating those properties with concrete instructions), and Social Impacts of computing (relating the implications and interpretations of computations to stakeholders and society). We use a "Data Science" context to motivate the learning experience, in contrast to previous approaches that focus on art and creativity. Our implementation of the course features both innovative technological scaffolding and modern pedagogical techniques (e.g., peer instruction, active learning, etc.) that enable us to capture and catalog a large quantity of learning analytics to guide course development. So far, we have offered this course over several semesters in a series of increasingly larger class sizes, iterating and improving the curriculum each time. Although initial results have been positive, we have identified a number of course components that we believe have untapped potential for making students more successful with regards to our learning objectives and motivational outcomes. In this extended abstract, we describe our experience of tracking learning analytics from a variety of sources over the latest offering of the course, and using the results of data mining to improve the courses' components, both pedagogically and technologically.

The course is taught in a collaborative, active learning style, with daily, hands-on activities that include conceptual work and concrete programming, all of which are systematically captured for analysis. Students are assessed at key intervals with pre-/post- cognitive examinations and motivational surveys. Scaffolded learning of programming through the use of carefully designed tools serves as the course's centerpiece, which yields fine-grained log data. In their first exposure to programming, students use a block-based environment (BlockPy) with automatic, guiding feedback. BlockPy features a dual, bidirectional blocks-to-text coding interface that is heavily instrumented, constantly recording user interactions and code. This environment facilitates the students' transition into Python during the last part of the course, yielding a more authentic programming experience. Throughout this process, students use datasets specially designed and scaffolded for introductory computing students with data drawn from diverse, real-world sources relevant to their diverse majors (e.g., supreme court decisions, historical slave sales, incidences of diseases across the country, real-time weather forecasts, etc.). The activities that students complete in the course, along with surveys, teacher observations, and interaction logs, provide a rich data source for our research questions.

Our final assessment of students is a month-long project where they individually apply Computational Thinking to answer questions relevant to their own major. Students use a dataset of their own choosing, write algorithms to process and visualize the data, and then create a detailed, video report on their findings. Students are evaluated using a rubric on the quality of their report across several metrics, including the quality of their questions, their ability to explain their data abstractions, and

the social impacts of their results. The programs that students create during this process are also evaluated, and the students are expected to explain automatically-extracted segments of their code using technical vocabulary. These two sources of data serve as ground truth of their Computational Thinking ability.

Our primary research question is to determine which analytics can be used early in the course to predict student performance at the end of the course with regards to our cognitive, motivational, and engagement objectives. With these predictions, we can answer our secondary research question: how one can design new interventions that can guide more students to success? These interventions can be remedial lessons that target particular misconceptions, recommended interactions with the teaching assistants or professors, and improvements to tools to enhance the support students receive. Our third and last research question is to determine what patterns of behavior are common among students in the course, and to identify and cluster students based on these patterns using data mining techniques.

These data inform the next iteration of our course lessons and tools. One of the biggest category of improvements are what automated tutor features can be added to the block-based programming environment to provide better guidance for beginner programmers—immediate, real-time feedback that can catch problematic behavior, correct students misconceptions, and forestall their frustration before it impacts their long-term course motivation. Another category is what types of data we are seeking to collect in future iterations, whether to rely more on course staff observations, automatic data logs, or the results of preliminary assessments. We will also identify relevant learning analytics that Computational Thinking courses can use to measure and cluster students early, thus enabling adopters to implement our curriculum at-scale effectively. Finally, we will identify common problematic patterns in students and potential solutions for supporting them that other tool developers could use to improve their environments.