Supporting Interdisciplinary Computing

Pedagogical Inquiry Grant Final Report

John Stratton: Computer Science, coordinator
Dalia Biswas: Chemistry
Claire Harrigan: Geology
Wally Herbranson: Psychology
Marina Ptukhina: Mathematics and Statistics
Jason Ralston: Economics
Peter Shultz: WCTS
Ashmeet Singh: Physics
Matthew Tien: Biology

Outcomes Overview

In our working group this semester, we discussed how we teach computational techniques to students across our disciplines, shared experiences and best practices, reviewed relevant pedagogical literature, and examined curricula from other institutions. We also sought input from students about their experiences. Based on our discussions, we plan to offer a CTL workshop for faculty in general about best practices for integrating computing into different disciplinary courses and curricula next semester. The immediate impacts have been primarily to our own current and future courses as individual faculty, but we will likely see more curricular innovations proposed and implemented over the coming years in response to what we've learned from each other and our students through this working group.

Pedagogical Strategies

We began our discussion by reviewing literature from within Computer Science Education on practices that the CS department regularly uses, including pair programming for in-class exercises and group projects.

Typical Learning Outcomes

In our discussions, we found that our disciplines often had some common learning high-level outcomes for students learning to use computational tools. Most participants agreed that in their courses, students should be able to:

1) Apply existing software tools for solving disciplinary problems
2) Adapt existing software to their specific context

3) Independently identify and troubleshoot software problems by finding and using publicly available documentation and examples

We hope that adaptations of these learning goals might be useful for more explicitly adding to current or future courses.

Learning Programming in Pairs

Pair programming is a common and well-researched approach in introductory CS courses to help students manage the cognitive load of learning to think about abstract problem-solving strategies and the specific syntax requirements of a particular programming language. Pair programming gives students problems to solve in pairs, with one student assigned the role of “driver”, person actually typing at the keyboard, and “navigator”, the person responsible for figuring out the abstract algorithm to implement and giving feedback to the driver. These roles swap periodically, and give students practice both with the mechanics of programming and algorithmic problem-solving skills but not overloading them with both in exactly the same moment. When done well, this can also help students manage imposter syndrome by seeing that their peers are also challenged, and create a supportive environment where students are actively helping each other succeed at the task at hand. Uncooperative students can make this strategy a challenge, but individual coaching on the benefits of prosocial behaviors to themselves and their peers can help students buy in more. While this approach is most often employed when teaching traditional programming languages, any environment where students need to simultaneously learn challenging abstract problem solving and concrete technology could benefit from this kind of approach.

Computing Tool Adaptation

Faculty from several disciplines outside of computer science found that trying to teach students a programming language from scratch is unnecessarily burdensome, and that students can often more easily achieve their goals in the discipline by adapting code written by others rather than learning to write new software from scratch. This does limit their ability to design novel computational tools for themselves, but is an excellent approach for building data analysis scripts or tools in many circumstances.

No-Coding Computational Tools

Many disciplines have computational tools with interfaces that don’t resemble classical programming languages. Discipline-specific “widgets” or well-designed user interfaces for specific problems are frequently introduced to students as a first exposure to computational methods or tools, particularly for students who may find math or programming intimidating at first. Many disciplines have some tools like this already available and can be accessible to both students and faculty with no prior computational experience.

Curricular Structures

In integrating computing skills into a department's curriculum, we found three common approaches within and beyond Whitman in a variety of disciplines.

1) Offer specialized courses dedicated to learning to use disciplinary computational tools
2) Supplement traditional courses with computational methods specific to those courses
3) Infuse computational methods throughout the curriculum, so that student computational skills are scaffolded along with their other disciplinary analytical and practical skills
4) Introduce new interdisciplinary curricula (minor or concentration) specifically on computing methods that can support a variety of existing disciplines

Each of these require a different degree of instructor and institutional commitment, but all are valuable, and each can be effective in achieving certain kinds of major learning outcomes. At Whitman, Chemistry is currently doing the most to attempt to integrate computation throughout the curriculum, with other departments using some mixture of the first two approaches. Clearly, larger-scale interventions have the greatest potential for more depth in computational skill building, but initiatives generally start with one or two faculty integrating computational tools into new or existing courses as a first step, which several departments are doing.

Potential Directions for Future Investment

The fourth option, creating a new interdisciplinary program for computing methods, seems interesting but premature, as we don't yet have enough pre-existing courses to draw from to fill out possible minor requirements that would be applicable to more than a couple departments, and the institution is still working out general principles for how a concentration could be defined. We identified some of the interdisciplinary course gaps, namely accessible introductory courses on data analysis and/or numerical methods from an interdisciplinary approach. Developing these courses would require more investment from departments and individual faculty, but could be very beneficial.

Multiple departments are in the process of adding more specialized courses on computational methods and/or integrating computation into more of the curriculum as a whole. There will be ongoing collaboration, for instance, as Computer Science will likely add the new Introduction to Computational Biology course as an alternative prerequisite to CS 167 for intermediate CS courses.

The Computer Science department is expecting to do a significant reassessment of its major structure and course offerings in the near future. Insights from this working group about the current gaps in accessibility of CS classes and transferability of materials taught in the current CS 167 course to other disciplines will contribute to those discussions.

Individual Impacts

John Stratton: Computer Science

What I have learned in this workshop will dramatically effect how I teach the next offering of CS 255: Computer Simulation Methods. In particular, I will use more instances of “copy and modify” for building basic simulations based on preexisting software, after seeing how effective that strategy can be in other disciplines. I will also be working with Ashmeet and Doug Hundley to better understand how my course and their courses on numerical methods complement each other. As the representative from Computer Science, I will also be taking all of these perspectives into the CS curricular review when it happens over the next few years, to better represent interdisciplinary concerns.

Dalia Biswas: Chemistry

This workshop taught me various pedagogical tools I would like to implement when teaching my computational chemistry/biochemistry courses. I used “copy and modify” Python coding techniques when preparing for different simulation methods in my computational biochemistry course. I will incorporate more in-depth Python coding, which seems to be a common programing language some faculty started incorporating in other science curricula (Ashmeet Singh in Physics and Matthew Tien in Biology). Students will be more equipped when they take my computational biochemistry course if Biology and Physics curricula introduce Python in some of their core courses.

I’m interested in exploring machine learning (ML) algorithms, which is a useful tool for Computed-Assisted Synthesis. Recently, I’m invited to join the NSF Center for Computed-Assisted Synthesis. I hope to expand my knowledge of ML and incorporate more projects in my computational chemistry course.

Claire Harrigan: Geology

If I were to teach Introduction to Geoscience Computing again, I would consider doing the course in Python to be in alignment with other science faculty, and I would try paired programming as a new learning strategy. One of the main things I got from working with the group is that there are students interested in scientific computing that could benefit from more structure in the curriculum and more discipline-specific courses.

Wally Herbranson: Psychology

I appreciated the chance to learn how computing is implemented in other departments, and how they compare to what is done in Psychology. In our most relevant required course, Psyc 210L, I plan to add in some subtle components that make computation more prominent (for example, by using syntax mode in Jamovi). I will also overtly connect the material to other courses outside

psychology (now that I'm aware of the landscape) so that students can more effectively seek out ways to strengthen their computational experiences after the class.

Marina Ptukhina: Mathematics and Statistics

Through our weekly meetings I learned a lot about programming/coding approaches in other departments at Whitman. One of the things that I'm excited to try out is the paired programming idea introduced at the workshop. It was also great to discuss opportunities that could be developed at Whitman to enrich student experience. Multiple ideas for new courses were discuss during our meetings that I believe are a very valuable outcome of this project. We hope to continue these conversations in more detail in the future.

Jason Ralston: Economics

I learned a lot about pedagogical techniques that other departments use during our meetings. I also learned about how other departments' faculties had come together to create learning goals specifically for computation. I am excited to try out some new ways of teaching programming in my econometrics course, and look forward to talking to my own colleagues about our goals for computational learning in economics.

Peter Shultz: WCTS

My participation in the group will enhance WCTS's ability to support computational pedagogy, particularly in departments outside of Math and Computer Science. It was an important chance to hear not just which tools (platforms, frameworks, languages) are currently in use, but also which are being considered for the future, and most importantly, what pedagogical values are guiding those choices.

Ashmeet Singh: Physics

My participation with the group has allowed me to better understand the computational landscape at Whitman, and will be instrumental in helping me integrate and embed computations in the physics curriculum. In particular, I am excited to try out pedagogical techniques, such as pair programming, that I have learned during interactions with the group in my upcoming course on Computational Methods in Physics which I am piloting in Fall 2023. I also look forward to continued collaboration with other faculty members across departments to better align our courses for a more interdisciplinary curriculum.

Matthew Tien: Biology

As a new faculty member teaching a new interdisciplinary subject at Whitman, I benefited greatly from the pedagogy discussions from the group. I was also able to identify the relevant faculty members at Whitman that can help me in future research projects. Moreover, I was able to understand how computing is conducted across other disciplines and how we may be able to

be more efficient in your pursuits of building computational methods into several curriculums across Divisions at Whitman.

Reflections

Overall, our results do line up with the high-level goals of our proposal, the most unexpected result being the conclusion that a generic interdisciplinary computing program is at best premature, with next steps better directed towards combined majors (e.g. a likely Computer Science/Chemistry proposal next year) and robust computing methods courses continued to be added to various other programs. We collected a body of valuable pedagogical materials used across multiple disciplines, with a draft of a CTL Workshop prepared that members will be happy to offer next semester. Those of us teaching this semester have been able to more thoughtfully assess our pedagogy in light of these materials, while others are planning to incorporate these pedagogical approaches into upcoming courses as indicated in comments above. The clearest outcome of the project is the sense of support that has been expressed by participants, particularly those in departments where they lack departmental peers invested in similar pedagogical goals of teaching computational tools. We look to ongoing impacts to student learning in the computational courses that will be offered in the coming years by the participating faculty, as indicated in the above comments.

The main limitation of this workshop is that the group of faculty was probably a bit too diverse for many of our topics to be universally applicable. While learning about the diversity of pedagogical goals was informative generally, it did lead to the fact that, for instance, our deeper discussions of pair programming simply weren't applicable to several faculty whose pedagogical goals in their courses didn't line up with the goals of that approach. Similarly, discussions of computing widgets didn't at all apply to some faculty. In retrospect, by identifying areas of shared pedagogical interest more quickly within our diverse group, we could have spent time more effectively by dividing according to pedagogical interests and spending time in smaller groups diving deeper into specific pedagogical techniques and strategies that would be more applicable to those within each group.