Computational Inquiry in Introductory Statistics

by Carl Toews

Abstract:
Inquiry based pedagogies have a strong presence in proof-based undergraduate mathematics courses, but can be difficult to implement in courses that are large, procedural, or highly computational. An introductory course in statistics would thus seem an unlikely candidate for an inquiry based approach, as these courses typically steer well clear of proof, have a list of required topics, and depend critically on computational software. On the other hand, the American Statistical Association (ASA) has long advocated the sort of active and exploratory class design that in many respects parallels traditional inquiry based learning (IBL). This paper reports on the author’s recent attempt to implement an inquiry based course in introductory statistics that fuses established IBL techniques with the specific pedagogical recommendations of the ASA. A signature feature of this course is that many of the inquiry-based activities are explicitly tied to computer work in the open source language R.

Keywords: IBL, inquiry, statistics, mathematics, education

1 Introduction
In the summer of 2014 I took part in an MAA sponsored workshop on Inquiry Based Learning [1]. While much of the workshop focused on paradigmatic applications of IBL within proof-based classes, the conversation also included adaptations for more procedural courses, such as calculus. Since a great deal of my teaching energy as a math professor at a small liberal arts college is devoted to first and second year courses whose content requirements are tightly prescribed, I was keenly interested in trying some of these adaptations in my own classes, and thus designed and implemented an IBL-like version of introductory statistics the following term.

While I had initially hoped that the IBL adaptations I had seen for calculus would carry over directly to statistics (which, like calculus, is at a lower mathematical level and is more procedural than most proof-based courses) the task was not as straightforward as I had hoped. To some extent the difficulties emerged from the enduring tension between mathematics and statistics. M. H. Moore, former head of the American Statistical Association, begins his article “Should Mathematicians Teach Statistics?” with a one word answer: “No!” [2]. Since my training is in mathematics, yet my professional duties require that I teach statistics, I routinely ignore Moore’s advice, but the reality is that statistics is born from data, not definitions, and thus it has a wholly different argumentative, analytic, and epistemic character. This is not to say that the cognitive and expressive end-goals that drive IBL in a pure math class are any less relevant in statistics, just that the way to achieve these goals needs to respect the origins and working methods of the field. One of the more tangible consequences of this recognition is the need to think through the relationship between IBL and the computer. In many mathematics classes, the computer plays an auxiliary role, but in a data-centric course like statistics, it needs to occupy a place that is front and center, which in an IBL setting means that computational activity needs to be one of the fundamental axes along which the inquiry is structured.

The purpose of this paper is to report on the particular class that these reflections gave rise to, and to use this report as the beginning of a broader reflection on the role of the computer within inquiry-oriented education. I argue that the strategic use of inquiry-oriented computer labs is not only compatible with the pedagogical agenda of the IBL community, but can produce important secondary competencies as well, especially when the computer work is done in an open-source, programming-centric language such as R or Python. Moreover, I argue that a particularly productive way to structure such a lab is to have students work in groups but each keep a physical laboratory notebook in which they record thoughts, conjectures,
results, and analyses. The dynamic interplay between the group and the individual, as well as the digital and the analogue, creates parallel productive spaces that reinforce one another and in which even weaker students can achieve a strong sense of voice and ownership.

The structure of the paper is as follows: in the second section I describe the nuts and bolts of how my class worked, framed within the setting of my particular institution and student cohort. In the third section I discuss outcomes, specifically as measured by student feedback, graded assessments, and my classroom observations. In the fourth section, I focus on the role of the computer within the IBL classroom, a discussion whose core is the idea of ‘computational thinking’ and whose scope extends beyond obviously ‘computational’ classes such as statistics. In the fifth section I introduce the notion of ‘parallel productive space’, and discuss the ways in which the interplay between physical and digital notebooks can help develop the ability to make and communicate scientific observations. The concluding comments of the last section include speculation about the potential scope of these ideas.

2 Class Nuts and Bolts

At the University of Puget Sound, introductory statistics is a multi-section, loosely-coordinated mathematics course that serves several audiences. As a mathematics course with limited prerequisites and a minimum of ‘mathematical’ content, introductory statistics draws many first year students who wish simply to fulfill their core mathematics requirement for graduation. It also draws students from applied fields like psychology and business for which some level of practical statistical literacy is methodologically important. Only rarely does the course draw mathematics or computer science majors, and for many students, this course is their only exposure to the mathematics department.

At our institution there is no separate statistics department, and the statistics teaching force is staffed by people with mathematics degrees, not statistics degrees. While this situation produces some predictable disciplinary distortions, anecdotal evidence suggests that it is common, at least in small liberal arts colleges like ours. Although there is agreement within our department on what topics should be covered, teachers have the latitude to design and implement their own sections, subject to meeting the content requirements.

I had two sections, each of which had 24 students and met four days a week with 50 minutes per class session. In concert with many (but not all) of my colleagues, I chose to use the text “Introduction to the Practice of Statistics” by Moore, McCabe, and Craig [3], a text that many students consider to be difficult but which has a good pedigree and strong structural support. I chose to divide the four class days into two types: Mondays, Wednesdays, and Fridays were regular days, and Thursday was a lab day. Homework was assigned daily each regular day, and collected the following regular day. Students did their lab work on their laptops in the open source language R, and for each lab, students submitted both their code and a physical laboratory notebook with sketches, results and analyses. I gave three mid-term exams, each of which had both in-class and take-home parts. I also gave a final exam, and had students complete a final project using data from the local public school system. Grading was apportioned among labs, homework, exams, the final project, and participation.

The dominant learning mechanisms in this class were the homework, in-class activities, and labs. The first two formed the core of the regular days. Each regular day would start off with student presentations of solutions to the previous day’s homework. Homework problems were “book problems”, generally between two to six exercises taken from the relevant section in our text. Some of these problems were procedural and some were conceptual, but for the most part they were straightforward, and designed to help students to consolidate the material. Students would volunteer for these presentations, but they had to complete a minimum of three presentations over the course of the term to get full participation credit. The presentations themselves were ungraded, but expected to be fluent and well-delivered. Non-presenters were encouraged to pose questions and comments, and to record notes or corrections directly on their own homework with special colored pens that I would pass out at the beginning of each class (an idea taken from Dana Ernst, who presented it at the MAA IBL workshop.) The annotated homework was then turned in and graded, where full credit was given for homework that had either been done correctly at home, or had been attempted at home, and corrected appropriately in class. (The colored pens made the distinction between these two things clear.) The remainder of the day generally consisted of my own brief overview of new material (5-15 minutes), followed by structured group activities that explored the concepts in greater detail and allowed students to get practice for the pending homework.
Laboratory days provided a marked change in pace. I chose to use R as the statistical language because it was free, powerful, and could be installed on students' laptops. (I was lucky: it turned out that every student in my class had a functional laptop.) Students worked in pairs according to a ‘driver-copilot’ model, where two students would share one laptop, and the driver would operate the keyboard while the co-pilot would review each line of code as it was typed in. Roles would switch half way through the class period. Every four weeks I divided the class into groups of four, and each week partners rotated among this set of four. The labs consisted of guided exercises that fleshed out and developed ideas that students had encountered during the regular days. In general, the exercises involved a combination of data analysis and numerical simulation, and were embedded in a PDF that I produced using Sweave, an interface tool for R and Latex. The PDF guided the students through the mechanics of code production, but also included ‘Pause for Reflection’ sections which asked students to reflect on results, make predictions, and generalize. (Example PDFs are hosted on the author's website, [4].) These reflections originated as discussions with the partner, but were recorded in physical laboratory notebooks, and these books were submitted each week for review, along with the code files.

Together, the lab days and the regular days gave students multiple learning modalities and exposure to both group and individual work across a range of computational and analytic problem environments. My decision to give two-part exams was an attempt to bring the distinct fruits of these different learning environments into an exam-level assessment structure. The in-class exams were timed, closed book, and done individually, thus closely mirroring the sorts of exams I gave in previous versions of this course, before my attempts to infuse it with an IBL flavor. The take-home exams focused on computation. Students were allowed to work in groups, but exclusively with their latest group of four, and the exam problems drew from coding tasks that this particular group had worked through together. Each student submitted his or her own code and analysis files.

The course concluded with a project. My goal was to give students an opportunity to grapple with real data in all its messy and unrefined overabundance. I chose to use local public school data as the basis of this project since it was publicly available and connected to the community in which the students were living. Data was broken down by school and included demographic composition, graduation rates, test scores, and levels of free lunch subsidies, among other things. Students were tasked to choose some subset of the data that they found interesting and to use it to tell a statistical story, replete with appropriate caveats and cautions. Students worked alone or in groups of up to three people, and submitted their work as a scientific paper that was assessed for both statistical content and expository quality.

3 Outcomes

I had three primary tools for assessing outcomes: student evaluations, graded assessments of student work, and classroom observations. My interest was in understanding the extent to which the innovations in this class strengthened or weakened student understanding, and also the extent to which they influenced student attitudes. Since the class was a one-off experiment with a single cohort of students, the outcomes are of limited generalizability, but can be understood as a “case study” which might inform subsequent investigation.

The student evaluations were very strong. Student evaluations at Puget Sound consist of 18 questions addressing issues such as course organization, level of challenge, interest value, etc. For each question students can write a comment, and also give a score between 1 and 5, with 1 being “poor” and 5 being “excellent”. The average student response over all questions was 4.4 for the “regular” version of the course I taught the previous year, and 4.8 for the “IBL” course, a significant increase. Moreover, the improvement extended to every facet of the evaluation: for all 18 questions, mean student response was higher in the IBL course than in the regular course. Qualitatively, most of the evaluations contained encouraging comments about the learning value of the class structure, pace, and environment. Many students felt that the labs served a particularly clarifying purpose, and a number of comments indicated that the labs formed one of the strongest elements of the course. There were some mild critiques about how much time it took to get up to speed on the computer, but overall students seemed to like the lab assignments. Some students objected to the class workload, but this critique was not a strong signal.

My graded assessments of student work were also encouraging. The in-class exams I gave were very similar to those I’d given previous years in my conventional version of the course, and provided a way of
assessing whether lost lecture time had adversely impacted student performance. The opposite turned out to be these case: the overall average for the IBL students was almost 2% higher. The higher exam scores did not translated into a shift in overall grade distribution, however, as the IBL course included a number of other assessment components, including labs, presentations, and a project. Students were kept aware of their overall class grade at all times. These facts suggest that the higher evaluations were not a consequence of students giving better reviews because they were getting better grades, but rather because they were authentically happier with the course and their learning.

Finally, my classroom observations provided strong evidence that the class worked well. I spent a lot of time at the beginning of the term building protocols for how to support one another during presentations and how to give constructive criticism, an investment that paid off with an exceptionally warm, congenial classroom in which the vast majority of students clearly felt comfortable. It took a while for some students to summon the courage to volunteer for their first presentation, but eventually all did, and I was struck by how some of the self-described ‘weaker’ students ended up delivering high quality reports. I was worried about possible student frustration with the R programming exercises, but instead, students seemed to find “lab day” one of the most enjoyable parts of the course.

4 IBL and the Computer

Many statisticians have argued that the real goal of the introductory statistics course is to cultivate “statistical thinking” [5, 6, 7], though some have claimed that this term is ill-defined and thus not a viable pedagogical target [8]. In their most recent GAISE report (Guidelines for Assessment and Instruction in Statistics Education), the ASA counters this line of thought by explicitly describing what it means to be “statistically educated”, coupling this description with guidelines on how and what to teach in an introductory statistics course [9]. These recommendations include the prioritization of concepts over procedures, the use of active learning strategies, the use of real data, and the use of technology.

While the first two of these recommendations are consonant with IBL objectives, the latter two are unusual within an IBL framework. One way to reconcile them is to conceive of the data and technology elements as gateways to a mode of thinking, specifically computational thinking. The Computer Science Teachers Association (CSTA) defines computational thinking [10] as “a problem solving process” whose characteristics include:

• formulating problems so that a computer can help solve them
• logically organizing and analyzing data
• representing data through models and simulations
• algorithmic thinking
• generalizing this process to other problems

The CSTA definition also notes that these processes are supported by a number of “attitudes and dispositions”, including confidence in dealing with complexity, persistence in working with difficult problems, tolerance for ambiguity, the ability to deal with open ended problems, and the ability to communicate and work with others to achieve a common goal or solution.

These “attitudes and dispositions” align closely with traditional IBL objectives. In an influential essay [11], Carnegie Mellon computer scientist Jeanette Wing makes the case that computational thinking is a “universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use,” and argues that such thinking has less to do with programming than with a conceptual, human mode of “sense-making”. Such sense-making involves knowing not only what can and cannot be computed, but also how to value and interpret such computation, a process that is often aided in a mathematical setting with experimentation [12]. In strikingly similar language, David Bressoud (former president of the MAA) describes IBL as a methodology whose foci include “sense-making, conjecture, experimentation, creation, and communication” [13].

The idea that the real target of technology in the classroom is computational thinking has implications for how it should be used as an IBL tool. The goal is not to produce technological sophistication per se, but rather access to a mode of thought. In my own labs, I tried to write “inquiry centric” tasks that guided students through computations, and then used these computations as a launching point for the kinds of thinking described above. Here are a couple of examples:
• In helping students understand the “68-95-99.7 Rule”, I had students generate 10 data points from a standard normal distribution, count the percentage of these points that lay within the interval $[-1,1]$, and comment on why the fact that this percentage is not 68% does not violate the 68-95-99.7 Rule. In trying to articulate this reason, students were forced to think through the notion of a sample, and to grapple with the idea of a “Rule” that was only true in some notional limit as the number of samples went to infinity. There were many ‘ah-ha’ moments with this exercise.

• In a similar vein, I had students simulate an experiment in which they flipped a coin twice and counted the number of heads. I asked them to think about performing the experiment 10 times. “Before generating any data, think about what the data will look like. Of the 10 runs of the experiment, what percentage of the results do you expect to be 0’s? 1’s? 2’s?” Their task was to sketch the predicted histogram in their lab book. They were then instructed to run the experiment and sketch the actual histogram, commenting on any differences between their predictions and their results. Once again, students got hands-on experience of the difference between theory and practice, and were compelled to articulate the sense in which theory and data were “consistent” in spite of visual evidence to the contrary.

• Some examples were more mundane. For example, the code snippet used to generate the above experiment looked like this:

```r
tossoutcomes = sample.int(2,2,replace=TRUE)-1
numberofheads = sum(tossoutcomes)
```

In my lab, I gave them the code, but then asked “Why do we subtract 1 in the first line of code? Why do we need the replace=TRUE flag?”. For students who have no experience coding and are not mathematically strong, these questions presents several challenges: understanding what the command `sample.int` does, understanding how to modify the existing routine to produce samples in \{0,1\} from samples in \{1,2\}, understanding the idea of a function with optional arguments. Students were required to make sense of the documentation and understand the connection between theory, code, and the computer.

Other examples can be found on my lab worksheets, available on the class webpage [4].

The broad push for technology in the classroom has implications for IBL educators. If the goal of IBL is to generate productive, empowered people in an inescapably ‘computational’ world, the students’ mental movement needs to encompass not just logical implication, but also order of magnitude, control structures, errors and approximations. And while sophistication with these concepts has implications that extend well beyond knowing how to get a machine to do anything specific, this does not mean that the machine is irrelevant: the computer continues to hold its place as the paradigmatic practice ground for computational thinking. In the case of my statistics course, the decision to give the computer a central role in the class and to use a programming interface made obvious sense because of the data-centric nature of the subject. But in retrospect, it was also an acknowledgement of the primacy of these issues, and as such, a paradigm that I will try to mirror in the future, not just in statistics but across the full range of my undergraduate courses.

5 The Role of the Laboratory Notebook

In their recommendations regarding technological exploration, the CUPM committee on technology notes that “to be effective, exploration must culminate in clear statements of what has been discovered” [14]. In my class, the way I enforced the production of such statements was through the introduction of the laboratory notebook.

Laboratory notebooks are standard fare in biology and chemistry, but I had never used them in a math class. In thinking about how to structure computer labs along IBL lines, however, the notebook emerged as an obvious tool. Notebooks serve several purposes, providing 1. a record of material covered, 2. a place to record graphical and numerical observations, 3. a place to articulate conjectures, and 4. a vehicle for communicating with others (and with oneself.) A notebook enforces clear and well organized draft-level expression, an objective consistent with IBL’s emphasis on “effective communication”. In my class, the notebook was designed to be an organic part of the way students synthesized the many heterogenous elements at play in statistics, and made sense of what the field was about.
I ended up conceiving of the notebook as a “productive space”, by which I mean a self-contained mode of production that has its own rules, standards of excellence, formal constraints and axes of creative freedom. In pedagogical terms I think of a productive space as linked to some particular activity (e.g. ‘homework’) but encompassing too the particular rhythms, expectations, and modalities that characterize that activity (e.g. ‘pen-and-paper, once-a-week, collaborative, low-stress.’) A rich classroom environment generally tries to make use of multiple productive spaces, and to ensure that there are structural and energetic synergies across them.

In this light, I think that the strength of the laboratory notebook was as a productive space that worked in parallel to several others, and managed to connect them in surprising ways. For example, writing computer code and completing paper-and-pencil homework exercises are, by themselves, both good exercises, but there can be a tendency for such assignments to live completely within the space in which they are assigned: computer work goes away when the screen is powered down, and one homework sheet has no interface with another. The laboratory notebooks pulled these productions into new contexts. Computer work was transcribed and interpreted with a tool usually reserved for homework (i.e. paper-and-pencil) while old homework was given new life as students used it to interpret what they saw on the screen. Lab book entries were predicated on discussion with partners, and thus drew on group modalities, but ultimately represented individual productions, and thus cultivated a sense of authorial agency. The power of the notebook emerged from its unifying centrality.

The extent to which students developed an authorial interest in their lab notebook was both a surprise and revelation to me. By taking in hand the same physical notebook week after week, students were able to see how their efforts were accreting to something large and complex. Faced with the inescapable reality that they were authoring some kind of a book, many students started to imbue their notebooks with their own voice and style. I commonly discovered voluminous and elegant reflections on, say, a histogram, that in any other context would have provoked at most a grudging sentence. Many of these book had a signature humor, a recurring marginal cartoon, a specific color scheme. Surprised by the quality, depth, and originality of these books, I realized that they had acquired a character that was both formal and informal, scientific and personal, computational and artistic. Perhaps by virtue of being so multifaceted, these books helped many students find a voice and acquire a sense of ownership over what might otherwise have been formidable foreign material.

In retrospect, I think these notebooks played an important role in giving my computer labs an IBL stamp. When I set out to design the course, I thought of the notebook as an expressive and synthetic tool, but it wasn’t clear to me how large a role it would play in my students’ learning process. I was happy to discover how powerfully the notebook fostered a many-sided engagement with the material, and thus how beautifully it played into the goals that lie at the core of IBL. By placing students in a structured but free authorial role, and by nudging them to use this role as a way of investigating many different modes of statistical understanding, the lab notebooks drew my students steadily down a path of inquiry, a path which led, in my view, to a much more sophisticated grasp of the subject than they would have achieved otherwise.

6 Concluding Remarks

Like many educational approaches, IBL is not a prescriptive set of techniques, but rather a philosophy and a stylistic orientation. While all IBL classrooms are student-centered, exploratory learning environments that prioritize clear arguments and strong communication, there is no core pedagogical method that makes a math class “IBL”. Nonetheless, there are certain canonical teaching techniques that show up again and again. Group work, presentations, and class discussions are examples of such canonical techniques, doubtless because they have proven effective in navigating the particular logical, aesthetic, and psychological demands of our subject. Computational inquiry, in contrast, still occupies a relatively marginal spot.

This article has sketched a scheme for an IBL-flavored statistics course in which computational inquiry played a key role. There are several issues to consider when thinking about how to extend these ideas to introductory statistics courses at other universities:

- **Scalability:** This course worked with a class size of 24. Grading laboratory notebooks was a major investment of time and energy, and in a class that was much larger, significant thought would need to be
given to how these notebooks would be assessed. The computational activities themselves could easily be extended to larger groups, assuming there were appropriate computational resources.

- **Hardware:** Each student in my class happened to have a laptop, but not every statistics class will enjoy this feature. Since students work in pairs, and only one laptop is needed per pair, the same paradigm could be used if only half the students in a class had laptops. But the model would still be less satisfying, as it would create a divide between those who “owned” the machines and software, and those who did not. Implementing labs on university managed computers would be a good option in this case.

- **Software:** This course used the programming language R, but many other computing environments could have been used to the same effect. I do feel that the open source element was empowering, as students could install the software on their own machines, and continue to use it long after the course was over. On the other hand, the benefits of this empowerment need to be weighed against the administrative hassle of installing the software on many different machines. Regarding computational interfaces, I feel that my choice to use a programming-centric one invigorated these labs. While menu driven statistical package may seem a safer choice for an introductory course, a command line or script interface probably plays better with the objectives of IBL and computational thinking.

The computational dimension in this course was “natural” in light of the course content, but the computational work was more than just a technological complement to the “real” work of inquiry: rather, it was a central IBL element in its own right. This observation suggests that similar kinds of computational inquiry might be fruitfully adapted to other IBL mathematics classes. A key element was that the computational labs required students not just to anticipate and parse computational output, but also to translate this output into clear expository language within their notebooks. Jupyter or Sage software notebooks combine executable code with typeset text, and thus offer the possibility of combining these elements. These open source software options would be excellent software choices for classes like linear algebra, differential equations, calculus, and numerical analysis. This same open source software has the flexibility to be used fruitfully in “purer” class such as number theory, real analysis, or topology, where it might be used with more emphasis on motivating or substantiating proofs.

For the would-be implementer of these ideas, it is important to recognize that there is considerable overhead in learning software and designing inquiry-based activities with it. I personally feel that the extent to which these labs energized and enriched the learning experience more than justified the work. Moreover, the work is easy to recycle, and can thus cut back on class preparation over the cycle of several years. Some purists might object that the use of software runs counter to our proud disciplinary paradigm of mathematics as a minimalist endeavor, one for which the only tools needed are a piece of chalk and an open brain. While many IBL models thrive in this minimalist environment, a computational paradigm is also possible, and might merit wider adoption as computational thinking plays an ever more important role in education.

The author would like to thank the editors and anonymous referees for valuable feedback on this manuscript.

REFERENCES


