# A Learner-Centered Approach to Teaching Computational Modeling, Data Analysis, and Programming

Devin Silvia[1(✉)], Brian O'Shea[1], and Brian Danielak[2]

[1] Department of Computational Mathematics, Science, and Engineering, Michigan State University, East Lansing, MI 48824, USA
{dsilvia,oshea}@msu.edu
[2] Fullstack Academy, New York, NY 10004, USA

**Abstract.** One of the core missions of Michigan State University's new Department of Computational Mathematics, Science, and Engineering is to provide education in computational modeling and data science to MSU's undergraduate and graduate students. In this paper, we describe our creation of CMSE 201, "Introduction to Computational Modeling and Data Analysis," which is intended to be a standalone course teaching students core concepts in data analysis, data visualization, and computational modeling. More broadly, we discuss the education-research-based rationale behind the "flipped classroom" instructional model that we have chosen to use in CMSE 201, which has also informed the design of other courses taught in the department. We also explain the course's design principles and implementation.

**Keywords:** Computational science education · Data analysis · Modeling

## 1 Introduction

In the last few decades, the use of data analysis and computational modeling has become critical to success in a wide variety of careers and the need for these skills is growing. As an example, McKinsey & Company recently released a report arguing that the need for data scientists – that is, workers who can successfully analyze, model, and interpret data, and use it to inform critical business decisions – is going to grow rapidly [8]. A wide range of industries now use computational modeling and data analytics to inform many aspects of their day-to-day practices, encompassing a range of activities that include product design, optimizing manufacturing processes, hiring decisions, and choosing advertising targets. The trend is clear, as are the implications for new employees – being fluent in the tools used to work with data and computational models, as well as being intelligent consumers of the products of these tools, will be a critical and high-demand skillset. Unfortunately, many institutions are relatively slow to provide students the type of computational training that addresses these needs, as indicated by the lack of programs aimed specifically at these areas. And, while not necessarily representative


J. M. F. Rodrigues et al. (Eds.): ICCS 2019, LNCS 11540, pp. 374–388, 2019.
https://doi.org/10.1007/978-3-030-22750-0_30

of all Computer Science departments, the courses offered by the Computer Science department at Michigan State University (MSU) are often heavily oversubscribed and typically do not encompass the range of data analysis and modeling skills that employers desire.

In 2015, MSU formed the Department of Computational Mathematics, Science, and Engineering (CMSE) to address the growing prevalence of computational and data science in academia and industry. The two core missions of the department are to perform cutting-edge, computationally-focused research and to provide a wide range of educational opportunities in computational and data science to MSU's undergraduate and graduate student populations. One particularly important population identified to be under-served in this regard by both the faculty and university administration is undergraduates in the natural and social sciences, as the primary avenues for computational training were computer science courses that do not address the needs outlined above.

In this paper, we describe the creation of CMSE 201, "Introduction to Computational Modeling and Data Analysis." This was the first course created by CMSE and is intended to provide a wide range of undergraduate students with the analytical and technical skills necessary to effectively work with data and computational models. In Sect. 2 we describe the design and implementation of this course. In Sect. 3, we summarize and discuss the limitations of this work as well as future plans, including challenges and opportunities.

## 2 Course Design and Implementation

### 2.1 Design Process

The broad goals of "Introduction to Computational Modeling and Data Analysis" (hereafter referred to as 'ICM') come from the process of creating the Department of CMSE. Prior to the MSU faculty endorsing the proposal to create a new department, a series of informal discussions were held with the deans, department chairs, and interested faculty in most of the colleges on campus. As a part of these meetings, faculty were asked "what sort of computational skills would you like your undergraduate and graduate students to have?" Disciplinary jargon aside, the answers were remarkably uniform: faculty would like students in their upper-division classes and in their research groups to be able to take a dataset and manipulate, analyze, and visualize it to extract usable information from it. Similarly, they would like students to be able to create simple models that capture the salient features of a system, and both quantitatively and qualitatively compare those models with the data that they have analyzed. The broad consensus among faculty was that there was not an existing undergraduate course at MSU that met most or all of these goals. A similar set of discussions took place with representatives from companies in a wide range of industries at several conferences (these companies were primarily in the manufacturing and high-tech sectors, and had a significant presence in the Midwest). These discussions had similar outcomes, though a much greater emphasis was put on the ability to *communicate* the process and outcomes of data analysis and modeling to co-workers and supervisors

(which is in line with more systematic surveys of the skills desired by employers) [12,13]. The skills identified in all of these discussions are the core of the high-level course learning objectives described below.

An additional goal of the design process was to create a course that exemplified the best practices as set forth in the undergraduate research literature. Our ability to do this was facilitated by several important factors. First, there are a large number of MSU faculty members who participate in disciplinary-based education research and/or successfully implement its outcomes in their classrooms, and thus there is strong support from both fellow faculty and university administration for the creation of such courses. Second, MSU has invested heavily in the creation of "REAL" (Rooms for Engaged and Active Learning) classrooms[1], which are specifically designed to facilitate student and faculty engagement and provide opportunities for innovative learning experiences. Finally, a key factor is that the course was created from a "blank slate," and was not a required course in any major or minor at the time of its creation[2]. This *ab initio* property of the course design meant that there were few institutional preconceptions about what should be taught or how it should be done.

Informed by the goals described above, we spent the Fall 2015 semester designing a new course and creating course content. Using the principles of *backwards course design* [18], we created both a high-level set of course learning goals (Sect. 2.2) and a much more finely-grained set of intended learning and content objectives (Sect. 2.4). We solicited feedback on these goals and objectives from many faculty and students, and the goals were iteratively tuned and improved. After solidifying our curriculum, we decided that we would evaluate student success based on their ability to work with data and implement and modify models. We decided on an assessment structure combining both formative assessment [2] (pre-class assignments and in-class group problem-solving activities) and summative assessment (homework assignments, exams, and semester projects).

### 2.2 Learning Goals

The over-arching purpose of this ICM course is to help students develop practical technical skills and ways of thinking that allow them to effectively construct, manipulate, visualize, and interpret datasets and computational models. These skills are not only applicable to the physical, life, and social sciences, but are rapidly becoming a necessity in these fields. In addition to developing technical skills and new thought processes, the course strives to promote an understanding of the social and scientific relevance of modern data analysis and computational modeling, which can be critical in recruiting and retaining students from underrepresented groups in STEM [11].

[1] https://tech.msu.edu/teaching/real/; see also Sect. 2.6.

[2] Although it is now a requirement in three degree programs and one minor, and a selective or elective course in several other degrees and minors.

As described in Sect. 2.1, this course targets students in the natural and social sciences rather than computer science students. As such, the computer programming skills that are needed within the course are motivated entirely by the goals and applications mentioned previously. Students certainly learn many of the same computer science topics (e.g., variables and arrays, loops, functions, etc.), but they do so in service of building and running simulations and analyzing data. Although this approach ensures that the students learn the skills necessary to ask meaningful questions of and extract answers from computational models and data, it places unavoidable limitations on the amount of computer science topics that can be covered in a single semester. More complex topics, like recursion and object-oriented programming, are left to a subsequent course for those students that are motivated to dive deeper. Finally, to reach a broad cross section of the student population, the only prerequisite for the course is one semester of calculus, and the course content touches on a wide variety of disciplines. We expect that our students have **no prior programming experience**.

The broad goals of the course are as follows. By the end of the semester, we intend that students who have successfully completed this class will be able to:

1. Gain insight into physical, biological, and social system through the use of computational algorithms and tools.
2. Write programs to solve common problems in a variety of disciplines.
3. Identify salient features of a system that can be codified into a model.
4. Manipulate, analyze, and visualize datasets and use to evaluate models.
5. Understand basic numerical methods and use them to solve problems.
6. Synthesize results from a scientific computing problem and present it both verbally and in writing.

### 2.3 Theoretical Underpinnings and Pedagogical Motivations

After developing the learning objectives for this course, the next steps were to determine the overall course structure, decide on the types of evidence that we would use to identify that learning was taking place, and sketch out specific assessments and assignments that would be used. While there are many theoretical frameworks that could be used to inform our design decisions, we decided to use the principles articulated in the book "How Learning Works: Seven Research-Based Principles for Smart Teaching" [1]. Our motivation for this choice is that the principles articulated in *How Learning Works* are grounded in a theoretical understanding of how people learn that is based on empirical evidence, and that these principles can be practically implemented in courses intended for large numbers of students and facilitated by instructors coming from variety of backgrounds and skill levels with only a modest amount of training. In designing this course, we have engaged most heavily with the following principles:

**To develop mastery, students must acquire component skills, practice integrating them, and know when to apply what they have learned.** In a given assignment, students are typically engaging with only a small number of

new concepts in programming, modeling, or data analysis. The arc of assignments focusing on a given topic (from pre-class, to in-class, to homework) are scaffolded to first give students practice with each new concept individually, and then integrate them with prior skills. For example, students learn to use iteration by first learning to write for loops in the pre-class assignment that perform simple tasks such as incrementing a variable. Then, in class they use the same conceptual structure to do numerical integration using the Trapezoidal Rule and, later, to evolve through time the position and velocity of a projectile acting under the influence of gravity and air resistance. As part of homework assignments and later pre-class assignments, loops are built upon as part of more complex modeling and data analysis tasks, often in scenarios where students have to make critical decisions about their usage.

**How students organize knowledge influences how they learn and apply what they know.** As described in the example above, new concepts are introduced in a scaffolded manner that assists students in making connections between new pieces of knowledge, and in productively integrating that new knowledge with existing ideas. The ultimate goal here is to help students develop a strong conceptual understanding of computational modeling and data analysis, as well as the tools and techniques for doing so, that they can apply appropriately in new situations. As an example of this scaffolding, students learned to work with variables prior to learning the concept of loops and their implementation, and in following activities learn Boolean logic (which is combined with simple loops to create more sophisticated control flow). After this, functions are introduced, and students integrate functions into loops and Boolean logic as they create increasingly complex models and data analysis workflows.

**Goal-directed practice coupled with targeted feedback enhances the quality of students' learning.** Each assignment addresses one or more course learning or content objectives. In-class assignments are typically pursued by students in small groups of 3–4 students, with instructional staff facilitating a handful of groups. Group members are encouraged to ask each other questions and the instructional staff regularly check in with all students to ensure that they are on the right track and to help guide the students in the right direction when necessary. Homework assignments and the semester project are accompanied by written feedback, with careful attention paid to struggling students.

**Students' motivation determines, directs, and sustains what they do to learn.** Students enter into our course with a variety of motivations, and are majoring or interested in a variety of topics. To that end, the topics of assignments are chosen to apply to a wide range of applications and subject areas, and to connect these areas together by the use of common numerical algorithms. Furthermore, we choose applications that are motivated by authentic scientific questions (e.g., disease spread) or social issues (e.g., racial segregation), and when possible, both (e.g., climate change).

**To become self-directed learners, students must learn to monitor and adjust their approaches to learning.** To this end, we structure assignments that are designed to encourage students to go beyond thinking about the

task that they are pursuing, but to think about *what they are doing*, *why they are doing it*, and *if they are doing it correctly*. We do this by asking students to justify the choices they have made and to explain how what they are doing fits within the context of the course.

Taken together, these principles motivate the course structure (as described in Sect. 2.4 through Sect. 2.7) and the course assignments.

### 2.4 Content Objectives

Working from our learning objectives and theoretical framework, we then chose the content of the course and its structure to achieve these objectives and to facilitate students learning the broad array of skills described in Sect. 2.1. The course progression is shown in Table 1, which includes a day-by-day breakdown of course activities. Motivated by our learning objectives, we describe a given day's activity in terms of the modeling and/or data analysis concept(s) that are developed during that class period, the context within which that is being taught (i.e., application area), and the new programming practice(s) that are required in order to be able to implement the modeling or data analysis concept.

The temporal progression shown in Table 1 demonstrates how students are gradually introduced to new skills and concepts. We assume that students come into the class with little or no programming experience, and thus early class sessions introduce fundamental programming concepts such as variables, loops, Boolean logic, and functions, and software-related tasks such as creating code flowcharts and writing pseudo-code to represent programs. These programming concepts are introduced using a modeling or data analysis concept that motivates the need for them – for example, in Day 2 of class students learn to use variables in programs to solve order-of-magnitude estimation problems. Motivated by the principles described in Sect. 2.3, we gradually introduce new skills and concepts in a way that facilitates their integration with previous content, and emphasize in class the ways that the new material builds on what has been previously learned. Furthermore, students are given clearly-defined problems in class, which they pursue in small groups with targeted instructor feedback. The modeling and data analysis/visualization concepts are taught within a variety of different contexts or application areas to both show the breadth of potential application areas and to engage students' personal interests.

### 2.5 Selecting and Training Instructional Staff

The instructional model used to support the learning and content objectives in this course is relatively resource-intensive and requires instructional staff to be carefully chosen and trained. We use a combination of PhD-level instructors, graduate teaching assistants, and undergraduate learning assistants. PhD-level instructors are assigned to the course based on interest and availability, and all faculty in the Department of CMSE are expected to rotate through the course.

**Table 1. Course Content and Progression.** An overview of the most recent iteration of the ICM course. The first column highlights the main modeling or data analysis concept explored in the pre-class/in-class activities. The second column provides the context in which the concept(s) was explored. The third column indicates the programming practices that were most important for the successful completion of the material.

| Day | Modeling/Data analysis concept | Context/Application | Programming practices |
|---|---|---|---|
| 1 | *Students are provided a course overview* | | |
| 2 | Order of magnitude estimation | Varied (e.g. population density) | Variable definition and manipulation, simple math |
| 3 | Mathematical representations of physical systems | Kinematics, projectile motion | Defining lists, writing loops, variable manipulation |
| 4 | Evaluating the state of physical systems | Kinematics, projectile motion | Writing loops, using boolean logic, using if/else statements, writing and using functions |
| 5 | Computing costs and optimizing solutions | Designing a ride share service | Writing and using functions, combining functions |
| 6 | Visualizing models | Projectile motion and population growth | Using Python modules (e.g. `math` and `Matplotlib`) |
| 7 | Loading and visualizing data | Water levels of the Great Lakes | Using the `NumPy` module, loading/reading csv files, saving plots |
| 8 | Numerical integration (finite different method) | Kinematics of extreme sports | Defining initial conditions, updating variables in loops, storing data in lists |
| 9 | Numerical integration (using pre-existing solvers) | Orbital Mechanics | Defining functions, using the odeint function from the `SciPy` module |
| 10 | Compartmental models | Epidemiology, rumors, population dynamics | No programming this day (instead: flow-charting and pseudo-coding) |
| 11 | Compartmental models | Epidemiology, rumors, population dynamics | Using `odeint`, modifying function parameters and initial conditions |
| 12 | Agent-based models | Forest fires and tipping points | Creating, navigating, and manipulating arrays, visualizing 2D arrays |
| 13 | Agent-based models | Schelling's segregation model | Creating, navigating, and manipulating arrays, visualizing 1D arrays |
| 14 | Data inspection, manipulation, and visualization | Flint water crisis data | Interacting with data using `Pandas`, applying boolean masks, data slicing and visualization |
| 15 | Data cleaning, transformation, and interpretation | Flint water crisis data | Manipulating data, computing statistical quantities, using `Pandas` |
| 16 | Finding trends (linear regression) | Political approval ratings | Using `NumPy`'s polyfit and poly1d and `SciPy`'s curve_fit to fit data, visualizing model fits |
| 17 | Finding trends (auto-regression) | Weather forecasting | Using the lag_plot function from `Pandas`, using fitting functions, visualizing data |
| 18 | *Midterm exam took place on this day* | | |
| 19 | Stochastic models using random numbers | Brownian motion/random walks | Using random number generators, implementing if/elif/else statements |
| 20 | Monte Carlo methods | The traveling salesperson problem | Using random number generators, visualizing data |

*continued*

**Table 1.** *continued*

| Day | Modeling/Data analysis concept | Context/Application | Programming practices |
|---|---|---|---|
| 21 | Markov chain Monte Carlo parameter estimation | Fitting noisy data | Using random number generators, evaluating conditionals, visualizing data |
| 22 | *Class time devoted to semester project work* | | |
| 23 | Manipulating and analyzing varied datasets | Climate Change | Loading data files, manipulating variables, visualizing data |
| 24 | Comparing models to data | Climate Change | Writing loops, storing data, loading data files |
| 25 | Building compartmental models, coupling models, and comparing to data | Climate Change | Manipulating and updating variables, visualizing data, over-plotting models and data |

Graduate teaching assistants (TAs) are chosen based on their expertise, availability, and interest in teaching with this instructional model. Undergraduate learning assistants (LAs) are students who took the course in a previous semester, received a high course grade, and have an interest in teaching.

Each week, the instructional staff meets to debrief their teaching experience, prepare for the coming week of material and, when time allows, discuss educational pedagogy. In the week preceding the start of the semester, all TAs and LAs are required to attend a full-day training workshop that explains the structure of the course, the motivations for the course design, common issues that may arise in class, and methods for effectively managing student-instructor interactions.

As the course has evolved, progressively more emphasis has been placed on ensuring that *all* members of the instructional staff receive sufficient training, including PhD-level instructors. As course enrollment continues to grow and our demand for undergraduates LAs increases, we plan to develop a more formalized TA and LA training program aimed at providing additional professional development and requiring that the students enroll in a formal course to explore educational pedagogies more deeply and present on their teaching experiences.

### 2.6 Course Technology and Logistics

In service of reaching our learning goals and implementing the described pedagogical techniques, we leverage a variety of digital and physical technologies. There are also certain course logistics necessary to achieve the quality standards that CMSE strives to uphold in its courses. Coordinating all of these aspects of the course are nontrivial and we attempt to outline the key details here.

Students in this course use the Python programming language[3] due to its ubiquity, overall ease of use as an introductory programming language, and the wide range of available and relevant libraries and supplemental software packages. Specifically, we have students install the Anaconda[4] Python distribution

[3] https://www.python.org.
[4] https://www.anaconda.com.

on their own laptop computers. Within this distribution, the main piece of software that we use in the ICM course are Jupyter Notebooks. In fact, the Jupyter Notebooks are a critical component of the course as they act as our student code development environment. They allow raw code, the results of its execution, and explanatory narrative to coexist in a single interactive document. The notebooks can also render plots and print data query results inline, making them extremely powerful for doing data analysis.

Jupyter Notebooks are also extremely powerful from a curriculum authoring and pedagogical perspective. We design our instructional materials with embedded YouTube videos of our mini-lectures and links to helpful resources all within a single document. That document then serves as the template that students edit and then submit for credit. This single-document structure means there is less pressure on students to context-switch between lecture notes, video demonstrations, and code playgrounds to try things out. Since notebooks support `<iframe>` cells, we can also embed a Google Form survey to solicit student feedback. Furthermore, Jupyter Notebooks are used both in academia and in industry, so students are working with the same professional-grade tools as experts.

Although students are expected to bring their own laptops to class, this expectation cannot always be met. We have two solutions for this: First, a cart containing laptops with the necessary software is brought to class every day for student use in class. Second, we maintain a JupyterHub[5] instance that allows students to remotely log on and use Jupyter notebooks from any device with a modern web browser. Between these two solutions, virtually any computing issue can be resolved with minimal impact on student and instructor class time.

Beyond the software that facilities the learning environment we strive for in the ICM course, the physical classroom space is crucial for motivating collaborative learning. To encourage productive collaboration, we offer as many sections of the course as possible in MSU's REAL classrooms, mentioned in Sect. 2.1 and shown in Fig. 1. These classrooms are constructed to discourage traditional lectures as students sit at tables in groups of 4–6 such that they are face-to-face with each other, rather than facing the instructor. These tables are equipped power outlets, audio/video cables, and hi-definition televisions, so that students can work on their laptops and share their screens to cooperatively code their problem solutions. Furthermore, the classrooms are set up such that should the instructor wish to share the work or progress of a particular group or individual, they can broadcast that screen to the entire room. This affords the instructor the opportunity to turn a localized learning moment into a classroom-wide discussion. The REAL classrooms also offer plentiful whiteboard space for student teams to write out their algorithm ideas, model components, and data analysis goals, an important part of the in-class activities outlined in Sect. 2.7.

Although the layout and technological enhancement of the REAL classrooms help to promote a collaborative learning environment, we note that teaching this course in classrooms like the REAL classrooms at MSU is not a requirement for course success. As long as the classroom desks are mobile, they can be

[5] https://jupyterhub.readthedocs.io/en/stable/.

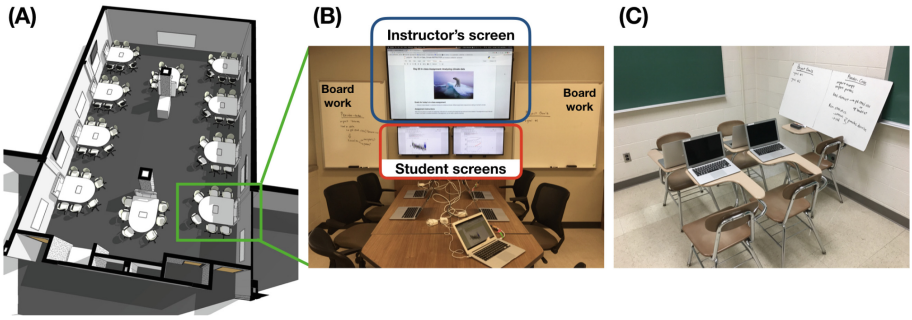


**Fig. 1.** (A) A rendering of one of MSU's Rooms for Engaged and Active Learning (REAL classroom). The room is designed to foster group work and minimize lecturing. (B) One of the REAL classroom student stations for group work. Students have space for their laptop, power access, and the ability to connect to one of the small screens at the end of the table. The instructor can display content on the large television screen above the tables to be viewed by all students. Students use the whiteboards for tasks like pseudo-coding, flowcharting, brainstorming. (C) One possible alternative setup for classrooms that do not offer the same design advantage of the REAL classroom. This setup achieves much of the same goals for encouraging student interaction and is sufficient for a successful offering of the ICM course.

rearranged into groups and the room can be supplemented with portable whiteboards. Although slightly less ideal, this allows for a setup like that shown in Fig. 1c, which still encourages peer-to-peer interaction during class.

During class students work in small groups for discussion, brainstorming, pseudo-coding, and other related activities, but occasionally fragment into pairs when writing code. Group members are expected to be mutually supportive, and are responsible for their own learning and each others' learning. They do so by sharing their ideas and reasoning, asking each other questions, and answering questions posted to them. A positive group environment is created in two ways. First, groups are deliberately constructed – instructors assign students to groups in such a way that groups have similar overall capabilities on average, that group members have a wide variety of majors, and that students from under-represented groups are not overwhelmed by those from the majority group. Groups are re-formed a handful of times per semester, with the instructional staff taking student personalities and abilities into account. Second, students are given clear instructions regarding the goals of the group and receive periodic feedback on how they and their group is performing.

Finally, the size of the course sections and structure of the instructional staff to date as follows. For the semesters discussed in this paper, course sections have included anywhere from 24 to 60 students. We strive to maintain a student-instructor ratio of no greater than 16:1, as going to larger ratios puts unrealistic expectations on our ability to provide sufficient facilitation of student learning [5,10]. For our current average section size of 36 students, we staff each section with one PhD-level instructor (typically a CMSE faculty member), one graduate

teaching assistant (TA) from the CMSE doctoral program, and one undergraduate learning assistant (LA). We maintain our desired student-instructor ratio in larger sections by adding additional LAs.

### 2.7 A Typical Week of Class

A typical week in ICM has the following structure. Individual sections of the course meet for 110 minutes twice a week. Prior to a given class, students are required to do a pre-class assignment distributed as a Jupyter Notebook. This assignment is short, due the evening before class, and typically includes narrative text and/or one or more short videos that introduce them to that day's new topics. After the students watch the videos, they immediately use what they have learned in a series of short programming problems and/or free response questions. Finally, they fill out an embedded Google Form survey that asks them to list any questions that they have about the material – as well as broader questions/issues they may have regarding the class – and then turn in the notebook via the course management system. The content of the Google Form is immediately available to the instructional staff in an easy-to-digest format which allows them to see what students are struggling with or have questions about. These pre-class assignments act as a mechanism for formative assessment and they are assigned grades based on completion rather than on correctness.

Class begins with a classroom-wide discussion of the pre-class assignment. This discussion may include a brief presentation by the instructor or a walk-through of particular pieces of the assignment. This affords the instructor the opportunity to address any questions or confusion that students may have indicated in the pre-class assignment, which helps put each student on equal footing prior to starting the day's activity. After this discussion, students download the in-class assignment from the course management system (another Jupyter notebook) and begin working on it in small groups. The assignments involve student discussion, calculations, programming, data analysis, and comparing of models to data, all of which is facilitated by the instructional staff. There are usually one or two "check points" during each class that are intended to get students to think critically about the outputs of their data analysis and/or modeling, and often result in whole-class discussions. These assignments also prompt the students to use the available whiteboards to write "pseudo-code," flow-chart their algorithms, or outline their data analysis methods. At the end of each class, students fill out a Google Form survey to give feedback and ask clarifying questions, and then upload a copy of their Jupyter notebook to the course management system. Grading is based on completion and good-faith effort rather than on correctness.

### 2.8 Summative Assessment

In addition to the various forms of formative assessments we described previously, we also build multiple mechanisms for summative assignment into the course. These take on the form of homework assignments, exams, and semester projects.

After students have explored a topic in class, they receive a homework assignment that builds on that topic, but typically focusing on a different application area. Students are encouraged to discuss the homework with each other, but are required to do the work themselves and turn in the homework individually. This homework is a summative evaluation graded on correctness. In addition to testing students' ability to perform data analysis and implement models, some homework assignments also have an interpretive or descriptive component. These questions provide us with opportunities to explore student understanding around topics that might be overly challenging to code.

Beyond the homework assignments, we have explored a variety of exam formats over the evolution of the ICM course. The most effective format involves giving students Jupyter notebooks that are similar in format to their normal in-class activities with a handful of questions aimed at probing key topics from the course. The questions asked might require: finding a solution to a set of equations to model a time-dependent physical system; loading, manipulating, and visualizing data; using random numbers to carry about numerical integration; or explaining what an agent-based model is, how it can be used to model a real system, and how one might setup the model using code. The exams reach a similar level of complexity as the tasks students normally complete in class, but for these exams they have to work individually. Students are allowed to use the internet to look up documentation for the code they are using or troubleshoot bugs they may run into. Students report that the exams are challenging, but fair and reasonably well-aligned with the course content and design. Averaged across sections, $\sim$80% of students achieve an exam grade of 70% or higher.

The final piece of summative assessment that we use in the ICM course are the semester projects. These projects provide the students with the opportunity to pursue a topic that they are personally interested in and showcase the skills and knowledge they have acquired over the course of the semester.

## 3 Summary and Discussion

This paper presents the design process, structure, content progression, and assessments mechanisms from the first six semesters of teaching and refining our ICM course. In the four years since the development of this course began in earnest, the Department of CMSE has grown substantially, as has student and faculty interest in the course. This presents both challenges and opportunities.

The primary challenge that we have encountered is rapidly growing demand for this course. Academic advisors and faculty are strongly recommending the course to students from a variety of majors, and enrollment has increased every semester, limited only by course capacity. Since the first offering, course sections have grown in size, with the most recent semester having a maximum section size of 72 students. These large sections are staffed by a faculty instructor, a graduate teaching assistant, and two undergraduate learning assistants. A similarly-structured introductory MSU physics course [9] suggests that is possible to scale

up to on the order of 100 students per section without sacrificing the student-centered nature of the course and its associated learning gains, something we will likely have to pursue.

In addition to growing demand from students, undergraduate minors and degree programs are now incorporating the ICM course into their requirements as a prelude to adding computational modeling and data analysis throughout their curriculum. Beyond the capacity increases required to support these programs, it will be critical to reassess the ICM learning goals and detailed curriculum to ensure that it continues to meet the Department's goals, as well as communicate this information to programs requiring the course.

The rapid growth in demand requires the development of a training curriculum for all instructional staff that focuses on methods for facilitating student-centered learning and the development of a more formalized undergraduate learning assistant program [15]. Undergraduate learning assistants are often comparably effective to teaching assistants [4] and may even help to institutionalize course reform and teaching practices [6]. For faculty-level instructors, training in research-based pedagogy is critical as most CMSE faculty are young, with little to no teaching experience and their experience has primarily been in traditional, lecture-style courses. Training faculty in the research-based teaching methods employed in this course is also crucial to its sustainability and continuity.

The success of ICM has led to inter-college discussions about the creation of other courses with complementary goals. For example, faculty in MSU's College of Communication Arts and Sciences are interested in developing a course focused more on data analysis/data science, the emerging field of "data journalism" [7], and the role that reliance on Big Data and algorithms can play in policy and broadly in society (e.g., [14]). The student population for this type of course would be quite different - journalism, marketing, humanities, and social science majors rather than STEM majors - which implies different course prerequisites and different learning goals.

The creation of this course (and its second-semester counterpart) creates a range of research opportunities. Critically, while a set of measurable concepts in programming has been defined in the computer science education research community (see, e.g., [17]), a similar inventory of concepts and a related assessment tool does not exist with regards to computational modeling and data analysis. Such a tool is critical for evaluating individual student assignments and in measuring the impact of curricular changes. Furthermore, the effect of the combination of computer science and modeling/analysis tasks on student learning has received almost no attention, outside of the physics education research community [3,16]. Finally, there is a rich vein of possible work relating to student affect and student identity. Most students who enter the ICM course have already declared their major, and are unlikely to self-identify as computational or data scientists. Exploring the impact of the ICM course on their self-identity and feelings about their major, particularly as they progress through their degree, may prove to be important for student recruitment and retention.

**Acknowledgements.** The authors thank Nathan Brugnone, Danny Caballero, Andrew Christlieb, Dirk Colbry, Sarah Gady, Nat Hawkins, Morten Hjorth-Jensen, and Luke Stanek for useful discussion and constructive criticism on drafts of this manuscript. We further thank all instructors of CMSE 201 for their enthusiastic participation and thoughtful feedback during the course creation process. We thank the MSU CMSE Department, the Office of the Vice President for Research and Graduate Studies, the College of Natural Science, the MSU Connected Mathematics Endowment (as administered by the MSU Program in Mathematics Education), and the Howard Hughes Medical Institute for their generous support.

## References

1. Ambrose, S.A., Bridges, M.W., DiPietro, M., Lovett, M.C., Norman, M.K.: How Learning Works: Seven Research-Based Principles for Smart Teaching. Wiley, Hoboken (2010)
2. Black, P., Wiliam, D.: Inside the black box: raising standards through classroom assessment. Phi Delta Kappan **80**(2), 139–144 (1998)
3. Caballero, M.D., Hjorth-Jensen, M.: Integrating a computational perspective in physics courses. ArXiv e-prints, February 2018
4. Chapin, H.C., Wiggins, B.L., Martin-Morris, L.E.: Undergraduate science learners show comparable outcomes whether taught by undergraduate or graduate teaching assistants. J. Coll. Sci. Teach. **44**(2), 90–99 (2014)
5. Cummings, K., Marx, J., Thornton, R., Kuhl, D.: Evaluating innovation in studio physics. Am. J. Phys. **67**(S1), S38–S44 (1999)
6. Goertzen, R.M., Brewe, E., Kramer, L.H., Wells, L., Jones, D.: Moving toward change: institutionalizing reform through implementation of the learning assistant model and open source tutorials. Phys. Rev. Spec. Top. Phys. Educ. Res. **7**(2), 020105 (2011)
7. Gray, J., Chambers, L., Bounegru, L.: The Data Journalism Handbook: How Journalists Can Use Data to Improve the News. O'Reilly Media Inc., Sebastopol (2012)
8. Henke, N., et al.: The age of analytics: competing in a data-driven world. McKinsey & Company (2016). http://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/the-age-of-analytics-competing-in-a-data-driven-world
9. Irving, P.W., Obsniuk, M.J., Caballero, M.D.: P3: a practice focused learning environment. Eur. J. Phys. **38**(5), 055701 (2017)
10. Kohl, P.B., Vincent Kuo, H.: Chronicling a successful secondary implementation of studio physics. Am. J. Phys. **80**(9), 832–839 (2012)
11. National Academy of Sciences, Committee on Trends and Opportunities in Federal Earth Science Education and Workforce Development, Board on Earth Sciences and Resources, Division on Earth and Life Studies: Preparing the Next Generation of Earth Scientists: An Examination of Federal Education and Training Programs. National Academies Press. https://www.nap.edu/catalog/18369/preparing-the-next-generation-of-earth-scientists-an-examination-of
12. National Association of Colleges and Employers: Job Outlook 2016: The Attributes Employers Want to See on New College Graduates' Resumes. http://www.naceweb.org/career-development/trends-and-predictions/job-outlook-2016-attributes-employers-want-to-see-on-new-college-graduates-resumes/
13. National Association of Colleges and Employers: Job Outlook 2017. http://www.naceweb.org/store/2017/job-outlook-2017/

14. O'Neil, C.: Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy. Broadway Books (2016)
15. Otero, V., Pollock, S., Finkelstein, N.: A physics department's role in preparing physics teachers: the Colorado learning assistant model. Am. J. Phys. **78**(11), 1218–1224 (2010)
16. Petter Sand, O., Odden, T.O.B., Lindstrøm, C., Caballero, M.D.: How computation can facilitate sensemaking about physics: a case study. ArXiv e-prints, July 2018
17. Qian, Y., Lehman, J.: Students' misconceptions and other difficulties in introductory programming: a literature review. ACM Trans. Comput. Educ. **18**(1), 1:1–1:24 (2017)
18. Wiggins, G.P., McTighe, J.: Understanding by Design. Association for Supervision and Curriculum Development, expanded 2nd edn. https://lccn.loc.gov/2004021131