

Motivating Introductory Computing with Student-Driven Datasets

Austin Cory Bart

September 12, 2015

Contents

1	Summary	
2	Problem Statement	1
3	Literature Review	1
3.1	Educational Theory	1
3.1.1	MUSIC Model of Academic Motivation	2
3.1.2	Situated Learning Theory	3
3.1.3	Contexts vs. Content	5
3.2	Introductory Computing Content	6
3.3	Introductory Computing Contexts	8
3.3.1	Abstract Contexts	8
3.3.2	Situationally Interesting Contexts	9
3.3.3	Empowered Contexts	10
3.3.4	Contexts that Make Instructors Care	10
3.3.5	Useful-Driven Contexts	10
4	Datasets as a Holistic Motivating Context	12
4.1	Data Science	13
4.2	Big Data	14
5	Existing work	15
5.1	Technical Infrastructure	15
5.2	Semi-Automatic Library Generation	17
5.3	Contextualizing with CORGIS	18
5.4	BlockPy: Block-Based Programming with Python and Data Science	19
6	Intervention Context	19
6.1	The Learners	20

6.2	The Content	21
6.3	The Course	23
7	Research Questions	23
7.1	Research Question 1: Technology	23
7.1.1	RQ 1.1: CORGIS Library Architecture	24
7.1.2	RQ 1.2: CORGIS Builder Architecture	25
7.1.3	RQ 1.3: CORGIS Gallery	28
7.2	Research Question 2: Impact	29
7.2.1	RQ 2.1: Motivation	29
7.2.2	RQ 2.2: Engagement	31
8	Work and Publication Plan	32
9	Conclusion	34
	References	34
	Appendix A Motivation Survey (Pre)	40
	Appendix B Motivation Interview	44

List of Figures

1	Content vs. Context	5
2	How to Add Fun to Education	6
3	A generalized model of Data Science[13]	13
4	The 3V Model of Big Data	14
5	RealTimeWeb Client Library Architecture	16
6	CT Students' Perceptions of Various Introductory Programming Contexts . .	18
7	CS Majors' Perceptions of the Benefit of Learning Different Technologies . .	18
8	Blockly vs. Python Code	20
9	Demographic Data for Computational Thinking Students	21
10	High-Level Course Overview	22
11	The same dataset can be structured differently according to the lesson at hand	26
12	The proposed CORGIS Dataset Library Structure	27
13	Students intent to continue their computing education vs. their self-reported motivation. Usefulness might be a better predictor than Interest or Caring (N=35).	33

1 Summary

This preliminary proposal defines a research plan to explore the problem of motivating introductory computing students. In particular, I propose that data science is an introductory context that can appeal to a wide range of learners across multiple dimensions of motivation. To test this hypothesis, I ground my work in two different educational theories – the MUSIC Model of Academic Motivation and Situated Learning Theory – to evaluate different components of a students’ learning experience for their contribution to the students’ motivation. In my literature review, I analyze existing contexts (such as game design and media computation) that are used in introductory computing courses, and their limitations with regards to my educational theories. I also analyze how data science has been used previously with only limited evaluation. I then describe my prior work and proposed future in providing facilitations to overcome technical and pedagogical barriers in using data science in introductory courses. I propose three new pieces of software that iterate on my prior work to solve the major existing technical challenges I have identified. I conclude by describing how I will evaluate this software and its impact on students motivation and engagement, and my timeline for doing so.

2 Problem Statement

Computing is increasingly considered a 21st century competency, leading to its growing entrenchment within universities' general education curriculum [61]. At Virginia Tech, for instance, the core requirements at the university have recently shifted to require all undergraduate students to take credit hours in “Computational Thinking” – emphasizing crucial topics in Abstraction, Algorithms, Computing Ethics, and more. The students in this course represent a diverse spread of majors from the arts, the sciences, engineering, agriculture, and more.

Although they have rich backgrounds, non-major learners are a challenge for instructors because they have no prior background in computer science, are not confident about their ability to succeed in the course, and have no assurances that it will be a useful experience.

These students are at-risk for dangerously low levels of motivation. As a more mature audience with domain-identified career goals and needs, existing interest-driven approaches to introductory computing may not be sufficiently effective, because students will not be convinced to learn material that doesn't have long-term benefit [25]. How do we motivate these unique learners?

This work will develop new pedagogical approaches and technical tools to better understand and resolve this question. In particular, I will explore the efficacy of Student-driven Data Science as an introductory context. This context moves beyond basic models of motivation to take advantage of multiple dimensions of engagement. Using complicated, real-world datasets is a known technique used in machine learning courses, and has even seen use in non-CS graduate level courses using the Software Carpentry method [60]. However, I propose new technology and pedagogy to enable its use in the introductory level. Further, I will conduct more comprehensive and fruitful evaluations than exists in the current literature.

3 Literature Review

3.1 Educational Theory

There are many elements of the learning process, and many theories of how people learn, how to teach, and how to engage with students. In this proposal, I investigate Academic Motivation - the process by which people choose to direct energy towards learning. Motivation is well understood to be a crucial part of education, particularly in Computer Science [10]. I apply motivational models against a learning model inspired by Situated Learning Theory in order to focus on the different aspects.

3.1.1 MUSIC Model of Academic Motivation

My work uses the MUSIC Model of Academic Motivation as its primary motivational framework. Although there are many motivational models available, few strive to be holistic models specifically developed for education. For example, theories like Expectancy-Value and Cognitive Evaluation Theory have a wider scope and have stemmed from other disciplines such as healthcare. The MUSIC Model is derived from a meta-analysis of these other theories, incorporating only the academically relevant components. Further, the MUSIC model is a tool meant for both design and evaluation, allowing it to be used in all phases of this work. Finally, the model and its associated instrument, the MUSIC Model of Academic Motivation Inventory (MMAMI), has been extensively validated and used in other educational domains, making it a reliable device [34].

The MUSIC model identifies five key constructs in motivating students [33].

eMpowerment: The amount of control that a student feels that they have over their learning (e.g., course assignments, lecture topics, etc.). This is amount of Agency that a student perceives.

Usefulness: The expectation of the student that the material they are learning will be valuable to their short- and long- term goals. There is no clear delineation of the time-scale for these goals, but there is nonetheless a distinction between strategic skills that students need to be successful in careers and personal interests and the tactical skills they need to complete present-day tasks.

Success: The student's belief in their own ability to complete assignments, projects, and other elements of a course with the investment of a reasonable, fulfilling amount of work. Most students desire experiences that are successful, but still challenging enough to be worth it.

Interest: The student's perception of how the assignment appeals to situational or long-term, individual interests. The former covers the aspects of a course related to attention, while the latter covers topics related to the fully-identified areas of focuses of the student. It can be difficult to parse out the difference between individual interests and long-term career goals – there can be alignment between these two components.

Caring: The students perception of other stakeholders' attitudes toward them. These stakeholders primarily include their instructor and classmates, but also can be extended to consider other members of their learning experience (e.g., administration, external experts, etc.). It can also be viewed as the extension towards society as a whole.

Students are motivated when one or more of these constructs is sufficiently activated. They are not all required to achieve maximal levels, and in fact that is not always desired – it is possible, for instance, for a student to feel too empowered, and become overwhelmed by

possibilities. For some of these constructs, a careful balance is required, and it may not be possible to ever achieve a **minimal** level; no matter how exciting you make your lecture, you may never convince your students it is interesting, although it is possible that they will still consider it useful and stay motivated. Crucially, students' subjective *perception* of these constructs is a defining requirement and is more important than objective reality. It does not matter if you believe that your lecture is **Useful**, if you have not convinced your students that it is (although an instructor's beliefs are a powerful tool for convincing their students).

The MUSIC model is often used as an organizational framework and an evaluative tool. As the former, it is a list of factors to consider when building modules, assignments, and content of a course. At all times, instructors can consider whether they are leveraging at least one construct to motivate their students. As the latter, it offers both a quantified instrument (MMAMI) and a structure to anchor a qualitative investigation on. The model has also been used in course design: Jones describes a controlled classroom experiment to motivate students by having an experimental group reflect on how a course satisfies the constructs of the MUSIC model. Students were simply prompted to answer the question “How will the material presented here be useful to you?”. Quantitative data gathered after the experiment indicated a significant increase in motivation [40]. The MUSIC model is therefore a flexible resource for studying motivation.

The MUSIC model has not been applied to Computer Science education before, although a number of other motivational frameworks have been leveraged in studies. Many of these studies often find results that hint at many of the elements of the MUSIC model. For instance, Mitchell describes interest, usefulness, and “intellectual challenges” (success) as three primary elements influencing success in computer science [41]. Although they do not discuss empowerment and social elements, many other studies often suggest these themes. Still, it should be acknowledged that it is an open question whether the MUSIC model can be applied directly, or if the domain has different characteristics from other disciplines.

3.1.2 Situated Learning Theory

A learning experience is a complex sequence of contextualized events, and Situated Learning Theory can help explain the structure of this experience — and in turn, this structure can be cast against the MUSIC model to explain the motivation over the course of a learning experience. Situated Learning Theory, originally proposed by Lave and Wenger, argues that learning normally occurs as a function of the activity, context, and culture in which it is situated [37]. Therefore, tasks in the learning environment should parallel real-world tasks, in order to maximize the *authenticity*. Contextualization is key in these settings, as opposed to decontextualized (or “inert”) settings. The key difference is that learning is driven by the problem being solved, rather than the tools available. Therefore, the problem being solved should lead directly to the tool being taught. This argument suggests that this is a key aspect of motivating students.

Authenticity is a crucial, recurring theme within Situated Learning Theory. All instruction and assessment must be aligned with reality such that success in the former leads to success in the latter. However, there is a subtle nuance here – authenticity is a perceived trait, not an objective one. Students derive value from their learning only if they *perceive* authenticity, regardless of whether the instructor has successfully authenticated the experience.

Situated Learning Theory has been applied to Computer Science before, with mixed results. A seminal paper by Ben-Ari [6] explores its application and limitations. This paper is somewhat hasty in its application of SL Theory by taking a macro-level view – they narrowly look to Open-Source and Industry Software Development communities as the only potential Community of Practice and interpret SL Theory as strictly requiring constant legitimacy, largely ignoring the possibility for gradual development of authenticity within individual courses and modules throughout a curriculum:

What I am claiming is that situated learning as presented in their work cannot be accepted at face value, because it simply ignores the enormous gap between the world of education and the world of high-tech CoPs, which demand extensive knowledge of both CS subjects and applications areas. This gap can only be bridged by classical decontextualized teaching in high schools, colleges and universities.



However, other researchers have found it a useful lens for analyzing curricula. For instance, Guzdial and Tew [25] used the theory to explore and deal with the problem of inauthenticity within their Media Computation project. SL Theory clearly has value, but only as a function of the way that it is applied. In this preliminary proposal, I use SL Theory as a generalized tool for exploring the topic of authenticity throughout introductory Computer Science.

The original work in Situated Learning Theory was not about pedagogy or instructional design. It described how people learn and the importance of context and collaboration, but it did not recommend a particular teaching style. Subsequent research by Brown [7] and others expanded the theory so that it could be applied to the design of learning experiences. These expansions often naturally dictate the use of active learning techniques, reducing the role of lecture in favor of collaborative, problem-based learning activities. Choi & Hannafin [12] describe a particularly useful, concrete framework for designing situated learning environments and experiences. The Choi & Hannafin framework has four key principles: the Context, described as the “... The problem’s physical and conceptual structure as well as the purpose of activity and the social milieu in which it is embedded”[50]; the Content, the information intending to be conveyed to the students; Facilitations, the modifications to the learning experience that support and accelerate learning (commonly done through Scaffolds); and Assessment, the methods used to evaluate the learning experience and measure the progress of the student. In my research, I focus on the value of the context and facilitations towards motivation, as opposed to the content and assessment.



3.1.3 Contexts vs. Content

There is a reciprocal relationship between contexts and content.

Figure 1 demonstrates an example of this relationship for the expected emphasis on big data as a context vs. content throughout an undergraduate curriculum, from a CS-0 (non-majors) course all the way to an upper-level course specifically on big data. Consider teaching Big Data as a topic: upper-level course could naturally have big data as a context and content, an introductory course would be unlikely to have it as content although it could be used as a framing story for assignments. Of course, many topics often bring some associated content with it. When students

learn programming in the context of, say, game development, they are almost necessarily learning content related to game development that may not be universal to computer science (e.g., how graphical resources are organized and accessed within the game engine). This content may be seen as a distraction by the instructor, or as useful side knowledge. For example, if a student had to learn how to use a command line in order to compile their game, they would be learning an authentic skill that might not be considered part of the core content, but is nonetheless generally useful. When evaluating a context, it is useful to consider what content it represents, and how authentic and useful it is. The authenticity of content that is attached to a context affects the authenticity of the learning environment as a whole. A good context enables a student to find recognizable elements and build on prior understanding, eventually being able to freely transfer their learning to new contexts.

Fascinatingly, the need for a strong context diminishes as learners mature and become domain-identified – the content itself becomes the context. Learners start to see other contexts as nothing more than distractions and unnecessary fluff. This makes sense – you would hope that Computer Science majors in their third semester would be naturally interested in the material, and this is borne out in experimental data. For instance, Yarosh & Guzdial attempted to integrate Media Computation in a CS-2 (Data Structures) course, and found that the learners had “outgrown the desire for a context” [62]. These results are similar to results we found in our interventions with a CS-3 course on Data Structures and Algorithms, where students seemed more irked by the surrounding context than intrigued.

Of course, it is up to the instructor to choose the depth and breadth of the context’s integration. The trade-off between the value and distraction added by the context is a delicate formula. Consider the scenario in Figure 2. A steel wall is a relatively relatable concept for most students – they can readily imagine such a large, durable object, and it somewhat

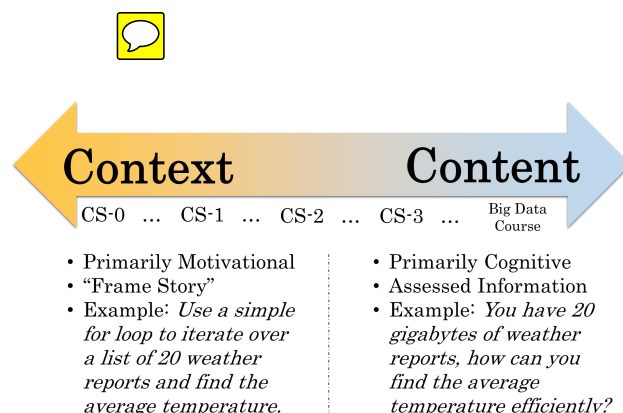


Figure 1: Content vs. Context

HOW INTRODUCTORY PHYSICS PROBLEMS ARE WRITTEN:

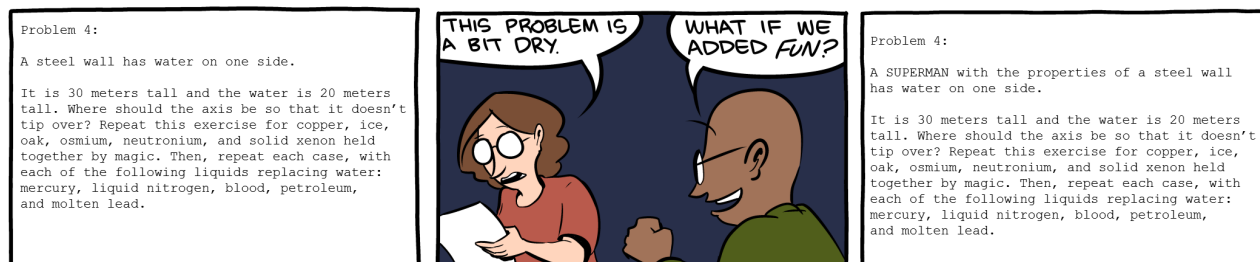


Figure 2: How to Add Fun to Education

Making the context “Fun” is not necessarily trivial, whether in physics education or computer science education. [59]

reasonable to expect that objects would interfere with it. In this scenario, the instructors consider replacing the wall with a comic book character – something that they anticipate will be more “fun”. If they are in tune with their learners, this might be an effective context – perhaps they know that their learners are comic book fans. However, because the integration is only at the surface level, it is possible that the learners will see this as a forced reference, and they will have a more negative reaction. It is also possible that they will not recognize the reference, or feel no positive emotions with it – many contexts do not take into account gender, racial, or socio-economic characteristics of the anticipated learner. I suggest that the process of motivating students using a context is non-trivial, and in the following section I will explore prior work in contexts for Computer Science Education.

3.2 Introductory Computing Content

Different Computer Science programs have different introductory curriculums, varying on whether they focus on Object-Oriented programming, Functional programming, etc. Complicating this discussion is the bifurcation of undergraduate introductory computing into Computational Thinking (sometimes referred to as CS-0) and Computer Science (sometimes referred to as CS-1), and the simplified curriculums used in K-12 education (e.g., the new AP CS course). However, **It is commonly** that any good introductory computing course will cover abstraction (representing complex phenomena more simply, usually as coded data), some form of decision-making (e.g., `if` statements, `cond` statements), and some form of iteration (e.g., `for` loops, recursion) [36, 45, 22]. From there, different curriculums lay out the material differently. The “How to Design Programs” curriculum emphasizes a functional programming model, using a LISP-descended language named Racket, and the only looping mechanism that students are taught is recursion. The dominance of the Object-Oriented Model in software engineering usually leads to a strong emphasis in introductory courses on abstracting data using Objects and Classes – such as the “Objects First” curriculum.



Besides the technical aspects, definitions include softer skills such as a tolerance for unstructured problems and collaborative attitudes [22, 31].

This proposal does not take a view on what should be included in an introductory experience, beyond a core of Abstraction and Algorithms. However, for practical purposes, materials and research work are grounded and tested in some course, and have adaptations for that content. Specifically, the proposed work is used in a Computational Thinking course, so that is how the content is oriented. “Computational Thinking” was coined by Seymour Papert in 1993 [46] and popularized by Dr. Jeannette Wing’s 2006 paper [61], which opened a floodgate of discussion about the term. Unfortunately, there is still limited consensus on *what* exactly CT is, whether it should be universally taught, how it should be taught, and how to identify when it has been taught.



An excellent resource for summarizing the history of Computational Thinking research is the 2013 dissertation by Wienberg [58]. This comprehensive survey analyzed 6906 papers directly or indirectly related to Computational Thinking from 2006-2011, describing research efforts and findings. Over half of the research on CT describes approaches to pedagogy (Curriculum and Program Description), leaving a small amount to modeling (Philosophy and Opinion) and assessment (Research and Evaluation). The lack of assessment research is understandable given the youth of this area of research, but still troubling. Even more troubling, however, is the further analysis of the 57 empirical studies. Only fifteen (26%) studies include or sought an operational definition of computational thinking, and only six go beyond the superficial (solely describing computational thinking as a “way of thinking”, a “fundamental skill”, or a “way of solving problems”). The failure to identify an operational definition weakens the theoretical strength of the studies. This weakness likely stems from the background of the researchers: only 18% of the articles involved education experts. In other words, over four-fifths of this educationally-oriented research appears to have been performed by people with no real formal training in educational research techniques. This is particularly troubling given that Computational Thinking is a strong target for interdisciplinary endeavors.

Weinberg reflects on the continuing debate about the importance of Computational Thinking:

Many, like Wing, believe computational thinking to be a revolutionary concept, one as important to a solid educational foundation as are reading, writing, and arithmetic (Bundy, 2007[9]) (Day, 2011[14]). Others believe its potential and significance are overstated (Denning, 2009[15]; Hemmendinger, 2010[29]), and some have voiced concern that by joining forces with other disciplines computer science might be diluting either one or both of the participating disciplines (Cassel, 2011[11]; Jacobs, 2009[32]). Both the praise and the criticism for computational thinking could perhaps be tempered by reflecting on a historical quote by Pfeiffer in 1962: “Computers are too important to overrate or underrate. There is no real point in sensationalizing or exaggerating activities which are striking enough without embellishment. There is no point in belittling either.” (Pfeiffer,

1962[48]).

Although it is ambiguous what Computational Thinking is, we will take it as a given that it requires students to learn some amount of non-trivial programming: using iteration constructs (e.g., `while`, `for each`, recursion, etc.), decision constructs (e.g., `if`, `unless`), have some sense of program state (through mutating variables or passed through composed functions), and require the programmer to translate instructions into a form the computer can understand. This is not meant to be a strict definition of everything a programmer should learn – simply an acceptable, minimal subset.



3.3 Introductory Computing Contexts

As part of the overarching goal to bring more students into Computer Science, a large number of contexts have been explored in Introductory computing. The context of a learning experience grounds the learner in what they already know, in order to teach the new material. Many introductory computing experiences focused on presenting the content as purely as possible, which can come across as abstract and detached [63]. However, starting with Seymour Papert’s work with robotics and the LOGO programming environment in the 70s [46], instructors have been interested in motivating students’ first experience with richer contexts. Some of these contexts rely on Situational Interest (e.g., Digital Media “Computation” (Manipulation) [23] and Game Design [63]), while others attempt to provide enduring career value (e.g., [Big] Data Science [2]) or short-term social applicability (e.g., Problem Solving for Social Good [24]). Ultimately, each of these approaches draws on different facets of motivation, they may be compatible with each other. In this section, I will discuss the implications of these different approaches.

3.3.1 Abstract Contexts


Denning describes the early perception of Computer Science by the public as “stodgy and nerdy” [16], since many early computer science classes were driven so strongly by mathematics and logic. A common early introductory programming problem, for instance, is writing a function to compute a Fibonacci number – a relatively simple task if you are familiar with the recurrence, and one that leads quite nicely to discussions on the implementation of algorithms, computational complexity, and a host of other subjects [42]. These contexts are “abstract” because they are already at a similar level of abstraction as the content they are attempting to convey. However, Oliveira [44] suggests that the discussion about “abstract vs. concrete” contexts is a misleading one, because the purity with relation to the content is less important than *prior knowledge*. According to modern constructivist and cognitivist theories, learners build on prior knowledge, and the ability to relate to what they know is crucial. The simple fact is that most students are not particularly good at mathematics, so



relying on it as a context is not a useful approach compared to finding subjects that students know and understand readily.

3.3.2 Situationally Interesting Contexts

As it became clear that Computer Science had a serious image problem, work began on making Computer Science “fun” and approachable. A key goal was to increase diversity and to broaden participation. This led to the rise of Situationally Interesting Contexts, emphasizing problems and projects that would be immediately appealing to a wide audience.

Guzdial, for instance, was largely responsible for the creation of the Media Computation approach, where students use computational techniques (e.g., iteration and decision) to manipulate sound, images, videos, and other digital artifacts. As an example, students might use a nested, numerically-indexed `for` loop in order to adjust the red-value of the pixels in an image, treating it as a two-dimensional array of binary tuples, in order to reduce the red-eye of a photo. 

Although wildly deployed, a review of these curricular materials by Guzdial [25] in light of Situated Learning Theory found that students did not find this an authentic context, and intense rhetoric was insufficient to convince them that it was authentic. Few students find it expedient and helpful to remove the red-eye from family photos by writing python scripts, and so are unconvinced that the context has long-term value to them (regardless of whether the content does). Guzdial leaves open the question of what contexts can be truly authentic for non-majors, given the relative novelty of teaching introductory computing for non-majors. Ben-Ari echos this question by suggesting a very narrow selection of authentic contexts and communities in his paper exploring the application of Situated Learning Theory to Computer Science in general [6]. Critically, the opposite problem could occur – if an instructor is effective at convincing students that a context is authentic, the students may believe the instructor even if the context is not authentic. There are serious ethical issues involved in misrepresenting the utility of a context, leading students to develop an embarrassing misconception of the field. Imagine a young child believing that all of Computer Science is game design, because that is what they started off doing.

There are other disadvantages of an Interest-driven approach. The motivation literature describes “Seductive Details” (interesting but irrelevant adjuncts) [27] as interfering both with short-term problem completion and long-term transfer. In other words, students get hung up on unimportant aspects of the context, so that they ignore the content. Consider a student using the game and animation development environment Scratch, which allows beginners to create sprites from images. A young learner might be so amused by the ability to change the color and shape of their image, that they neglect their assigned work. Although a well-regulated learner would not be distracted, most of the at-risk population that would benefit from these contexts are unable to deal with such distractions. Of course, this could be said of any context, but there is particular danger from a seductive context.

Kay [35] identifies another, potentially critical problem of relying solely on situationally interesting contexts, particularly when it leads to individualized interest towards that context and not the content. What happens once a student has completed the introductory course and is ready to move onto further courses? Most later courses are more decontextualized, and will not use contexts such as game development, robots, etc. Kay goes so far to say that it is unethical to suggest to students that a contextualized introductory course is representative of the curriculum as a whole.

3.3.3 Empowered Contexts

Orthogonal to the idea of making a context fun is the idea of giving students more freedom and agency to control their learning. Compared to Interest-driven contexts, this approach is comparatively less researched, although it is not an uncommon practice. Instructors will often allow students to choose from a range of projects or assignments. Stone [53] ran a 2-year study where students were allowed to choose their projects from a wide range of domain areas (e.g., Biology, Math, Business, Etymology), and were then surveyed on their engagement. Unfortunately, their experiment suffered strongly from low enrollments and even lower survey responses. Therefore, it is difficult to believe any of their results (including the idea that women are more likely to prefer biological- and meteorological-themed projects). However, their experiences do suggest an interesting challenge: normalizing the difficulty (both in terms of computational knowledge and domain knowledge) across many different projects is a struggle.

3.3.4 Contexts that Make Instructors Care

Most modern educational theories argue that learning is inescapably affected by social factors. There is evidence that the instructor [55] and fellow students [3] are the most important factors in an introductory experience, for instance. Kay [35] discusses this explicitly. It is possible that the most important element in a context is not whether it is fun or useful, but whether the instructor can get excited about it and impart that enthusiasm to the student. And not just enthusiasm, but a thorough understanding of the problem, its usefulness, and the rest of its attributes.



3.3.5 Useful-Driven Contexts

An alternative focus to Interest is Usefulness, the idea that the context should have immediate or eventual benefit to the learner's needs. To some extent, it is impossible (or at least prohibitively difficult) to find a one-size-fits-all context that will be useful to all learners (designing learning experiences without your learners in mind is an example of preauthentication [47]). The ideal situation for any instructor is to create contexts that specifically suit

the interests and values of your learners [18]. However, in practice, some contexts are broadly useful and are likely to engage a diverse crowd of learners. In this section, I suggest two distinct contexts that might fall into this role: real-world problem solving and data science.

In theory, Computer Science provides tools for solving problems, and it is possible that the problem solving can be done with even the simplest tools [38, 24]. An ITiCSE Working Group collaborated to produce a new framework centered around these ideas – “Social Computing for Good”, a collection of approaches and projects for interdisciplinary students to solve using computing [24]. They raise a number of issues with using socially relevant materials: that games and graphics can appeal to instructors as a “cheap” source of motivation, that students and instructors can become cognitively overloaded by the addition of domain knowledge, and that instructors can even be intimidated if they don’t have expertise in the domain area. The working group also create a valuable rubric for developing and evaluating problems (which I map to elements of the MUSIC model below):

1. The degree to which the problem is student-directed (eMpowerment)
2. The amount of scaffolding needed (Success)
3. The amount of external domain knowledge needed (Success)
4. The contribution to the Social Good (Usefulness, Caring)
5. The “coolness” or “sexiness” (Interest)
6. The amount of explicit student reflection incorporated (Usefulness)



Although this framework presents some ideas, there are still unsolved technical and pedagogical problems in how to optimally bring these materials to learners. Their paper ends by raising questions about the effectiveness of this approach compared to existing methods.

A number of other researchers have created course materials along similar lines. Erkan [21] had a sustainability themed curriculum – student surveys suggested some level of effectiveness, although the sample size (N=16) was far too small to generalize the results. In addition to providing two case studies, Buckley [8] suggests an interesting delineation between Social problems (uppercase S, indicating problems general to society) and social problems (lowercase S, indicating problems of personal interest to the learner).

Barker ran a large survey asking why students persist towards majoring in CS [3] (N=113, only freshmen) and performed a factor analysis. Critically, they found that Meaningful/Relevant Assignments (subsuming both Interest and Social Usefulness) were a major factor in whether students would persist in the major – however, it wasn’t one of the primary factors (that is, students suggested other factors were more important for deciding whether they would persist). Interestingly, whether students felt that their workload and pace was appropriate was a much bigger source of importance, suggesting that care and attention should

be given to making a context suitably difficult before it is made interesting and useful. This focus on ensuring normalized, appropriate difficulty is echoed by several other authors, all of whom suggest that doing so is not trivial [49, 52].

In the past two decades, the field of Data Science has emerged at the intersection of Computer Science, Statistics, Mathematics, and a number of other fields. This field is concerned with answering real-world problems through data abstractions, and offer a less socially-conscious path to Usefulness. As a context, there are pedagogical penalties for using it, since it introduces a wide variety of new content including visualization, statistics, ethics, and social impacts [1]. However, a good instructor can downplay the focus on these side-areas as needed, or even emphasize subject matter’s strengths (e.g., a statistics major might find it interesting to use their mathematical background to strengthen their problem-solving investigation). However, there are other difficulties. Bringing in messy data requires real sophistication by the instructor, especially when working with Big Data.

The use of data analysis as a form of contextualization is not novel, and represents a new and actively growing movement where instructors create programming assignments with specific datasets in mind [2, 54, 26, 17]. Upper division courses have employed these situated learning experiences using data of varying size and complexity for several years [20, 56, 57]. However, in all of these research papers, there is typically little evaluation of the advantages and disadvantages of using data science in introductory education. Although Sullivan [54] did conduct a study on the difficulty and usefulness of the datasets they provided, they do not go far in identifying lasting lessons for educators creating such datasets. Other researchers conducted even less impressive studies: DePasquale [17] included exactly ONE student response in their evaluation.

4 Datasets as a Holistic Motivating Context

I propose using datasets as a motivating introductory computing context. **My core thesis is that data science, used properly, is an excellent context for motivating students across multiple components of motivation: providing opportunities for agency, a sense of usefulness and authenticity, controllable complexity, the potential to be interesting situationally and towards individualized interest, and a promotion of caring and ethics.** Data Science accomplishes this by being a “meta-context” – it is a general framework for working with an infinitely wide array of different data contexts. However, there are many challenges in doing so – managing data is tricky and can be a very unstructured problem. For introductory students, in fact, it can be too overwhelming to find and dive into an arbitrary dataset, requiring significant scaffolds to get anywhere.

As established in the previous subsection, this is not a novel approach – upper forms have always historically used datasets in courses on Machine Learning and Database Design, and recently lower forms have started too [2]. However, there is very little evaluation of the

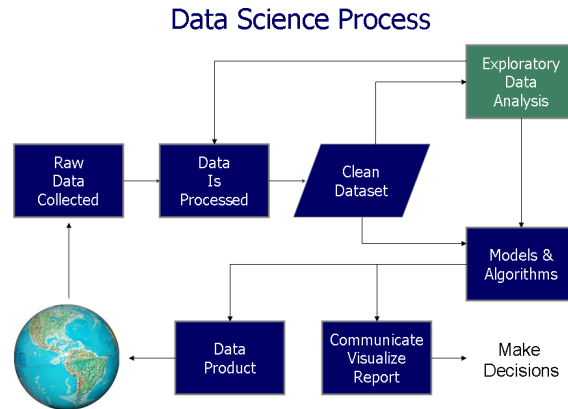


Figure 3: A generalized model of Data Science[13]

success of this approach, and numerous roadblocks exist in using this context. For my dissertation, I will investigate what technology can be used to take full advantage of this context according to a number of constraints. Before describing what I have done and will do, however, this section will describe this context in more detail.

4.1 Data Science

Data science is the process of answering questions by building, exploring, and processing datasets. There are many theoretical models that define the term more strictly, but in general it is described as an iterative model of collecting, sanitizing, processing, rendering, and interpreting. Figure 3 gives a visual overview of this process. My research will not attempt to narrowly define data science; instead, the goal is simply to use elements of the data science process to contextualize the experience of learning about computing topics such as abstraction and algorithms.

It is crucial to understand that the goal is not to teach students how to become data scientists, anymore than it is the goal of Media Computation to teach students how to be professional computational artists. As a context, using datasets is simply a means to an end – it may involve any components of the generalized data science process at any level. Some professors may identify data science as a learning objective in and of itself. That is fine, but they are attempting to teach something that is, at some level, distinct from computing itself.

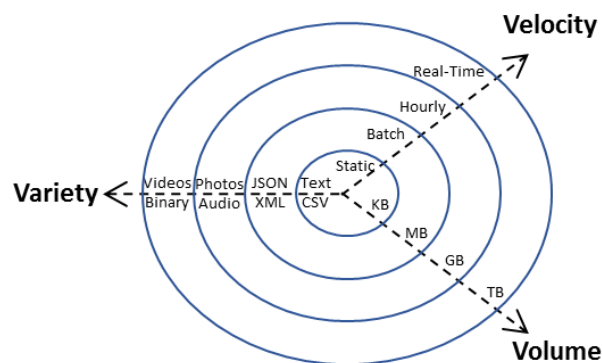


Figure 4: The 3V Model of Big Data



4.2 Big Data

One particularly important subtype of datasets is known as Big Data. Big Data is much in the news these days, from reports of massive data dredging by the NSA to tens of millions of credit cards stolen by hackers from commercial databases. Big Data has become crucial to scientific advances from understanding the genome to predicting climate change. Even more so than regular data science, there are many obstacles to effectively educating students on Big Data. Its representation, manipulation, and expression is, by definition, challenging, and modern curriculum and programming tools are typically inadequate.

Big data has been loosely described as quantities of information that cannot be handled with traditional methods [39]. But “traditional methods” is a vague phrase that has different meanings to different learners. To a Humanities major in their first CS-0 course, the traditional method to sum a list is to use Excel. In this scenario, “big data” means anything that won’t comfortably fit into Excel’s working memory. However, to a third-year Computer Science major, the traditional method would be to write an iterative or recursive sequential loop; being given big data forces them to explore parallel models of execution. Clearly, “big-ness” is a function of the learner’s experience, but that is still not a solid definition. A more precise definition is the “3V Model” [19], which posits that there are three dimensions that distinguish big data from ordinary, run-of-the-mill data:

Volume: The total quantity of the information, usually measured in bytes or number of records. However, this also extends laterally: the number of fields in the structure of the data also impacts the complexity and size. The threshold at which data becomes big is a function of the hardware and software being used. For instance, embedded systems may consider gigabyte-sized files to be big, while modern servers might not struggle until the petabyte level.

Velocity: The rate at which new information is added to the system. High velocity big data

implies a distributed architecture, since new data must be arriving from somewhere. The dynamicity of data can vary widely across these architectures, with data updating every year, every day, or even multiple times a second.

Variety: The format or formats of the data. Ideally, data are always distributed in a way that is readily accessible. For instance, simple text-based formats such as CSV and JSON are widely supported, relatively lightweight, and human-readable. More sophisticated data formats for image and audio are also typically well-supported, although still more complicated. However, projects using specialized, compressed binary formats or, more dangerously, multiple formats (e.g., image archives organized with XML files), are more complex.

Silva [51] taught an introductory course truly focused on techniques for tackling Big Data: NoSQL, MapReduce, NewSQL. Unfortunately, they did not conduct any kind of evaluation of their work across any of the expected dimensions. Learning to work with Big Data can add extra authenticity to the context, but it also raises a large number of new challenges. Once again, it is not the goal of my research to teach students how to work with truly Big datasets. Instead, I will explore whether the use of Big Data is a useful means to motivate students.

5 Existing work

As I am now three years into my PhD, my research builds on significant prior work. First and foremost, my CORGIS project has already been used in several introductory programming experiences, and seen publication at multiple venues, including two successful SIGCSE workshops. My primary research project for the first two years of my dissertation led to CORGIS: Collections of Real-time, Giant, Interesting, Situated Datasets. This project's goal is to make simplistic data sources available to learners early in their programming experience, so that they can explore Big Data Science contexts.

5.1 Technical Infrastructure

The foundation of this proposal rests on prior work developing the RealTimeWeb project, a software architecture framework that provides introductory programming students with an easy way to access and manipulate distributed real-time data[4]. Real-time data is a specific branch of Big Data – specifically, high velocity data.

As **our** focus shifts from real-time data to datasets in general, we have renamed our overarching project from RealTimeWeb to CORGIS. Our work is now available at <http://think.cs.vt.edu/corgis/>. As a successor to the RealTimeWeb project, CORGIS retains all the



previously developed libraries for accessing high velocity data; however, it also contains our new libraries for working with high volume data. These libraries are paired with potential assignments and helpful documentation for deployment. Long term, we intend to gather and disseminate data on the success of these libraries, with the hopes to establish a community of developers and educators that create new resources. For clarity's sake, we use the name RealTimeWeb to refer to the architecture we have developed to rapidly connect to high velocity data streams.

At the heart of our project are carefully engineered, open-source client libraries through which students can access the data provided by real-time web services. We provide client libraries for a number of data sources, such as business reviews from Yelp, weather forecasts from the National Weather Service, and social content from link-sharing site Reddit.com. Each of these client libraries is in turn available for three common beginner languages: Java, Python, and Racket. These libraries do more than just streamline the process of accessing distributed data, however. Each library is built with a persistence layer that enables the library to work without an internet connection. Not only does this ensure that students without a solid internet connection can maintain productivity, it also simplifies developing unit tests. In fact, this technical scaffolding for the students circumvents most of the difficulties of distributed computing, including HTTP access, data validation, and result parsing. Figure 5 demonstrates the architecture used in our libraries.

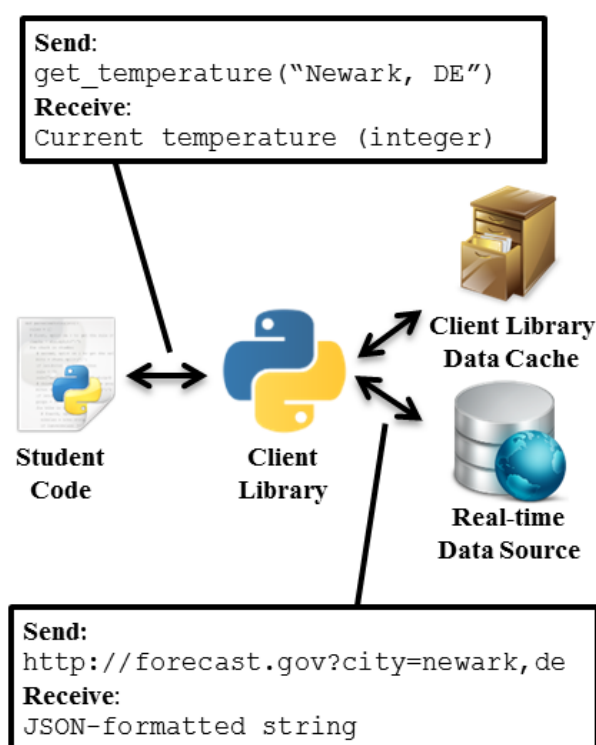


Figure 5: RealTimeWeb Client Library Architecture

The persistence layer is implemented using a caching mechanism, but not a traditional one. In a conventional caching system, the result of every call made to the external service is memoized using a key-value store, often with a timestamp in order to expire out-of-date data. In our caching system, an instructor preloads the cache with a sequence of data values for each expected call to the external service. Although this limits the number of possible calls to the external service, it improves the consistency of the experience. Instructors can specify policies for how the system returns data – if the cache runs out, it could restart with the initial result, a developer-specified “empty” result, or repeatedly return the final result. For example, consider the United

States Geological Services’ Earthquake data stream, which exposes a function to retrieve a list of earthquakes around the world for a given time period (e.g., past hour, past day, past week, etc.). The instructor could provide a cache to simulate a period of high seismic activity, returning a large number of earthquakes every time the function is called. If the user exhausts the data in the cache, it might be programmed to return an empty list, signaling no further activity.

Our High Volume libraries often work by providing sampled versions of the data internally, so that students can work with faster, representative subsets. Then, when they are ready for full deployment, they can switch to a “full production” mode – trading speed for quality. Most of the work in developing these libraries is organizing the sampled data to load into memory and get processed as quickly and naturally as possible.

An alternative scheme for distributed High Volume libraries uses a more traditional caching strategy – assuming that the data source is largely static, requests are cached locally. This caching is typically done using a simple key-value store on the local filesystem. Limits can be set on the size or lifespan of the data in the cache, allowing the system to update itself according to the velocity of the external data service. Using open-source REST API generating tools such as Eve [30], existing high volume datasets can be quickly transformed into distributed datasets, alleviating the issues of data storage and transmission.

5.2 Semi-Automatic Library Generation

Our client libraries are easily available through a curated, online gallery; each library is designed to be quickly adapted to instructors’ specific academic desires. This gallery also provides a tool for rapidly prototyping new libraries based on our framework. As an open-source project, we encourage collaborators to explore and extend the tools that we have created.

The process of connecting to online data sources is fairly uniform: performing an HTTP request to a URL returns formatted data (typically JSON or XML). This information must then be parsed into native data structures, and then filtered and transformed into an appropriate domain object using the language’s proper construct (e.g., structs for Racket, classes for Java). We have established a JSON-based client library specification file format that can be compiled to appropriate source code in each of the three target languages using the modern templating language Jinja2. This compiler can be easily extended to new languages by providing a template. This specification format has two major components: the functions, which connect to the relevant URL endpoints, and the domain objects, that are generated from a successful HTTP request. This open-source tool has been used to successfully and rapidly develop a number of the existing client libraries.

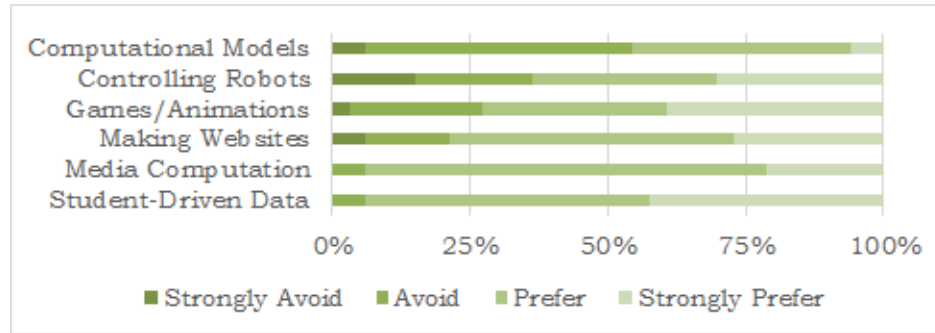


Figure 6: CT Students' Perceptions of Various Introductory Programming Contexts

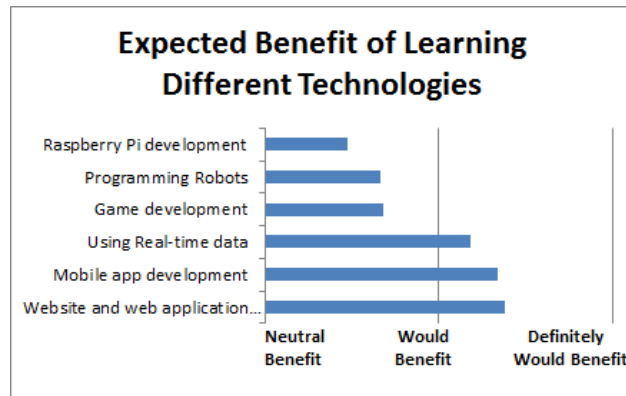


Figure 7: CS Majors' Perceptions of the Benefit of Learning Different Technologies

5.3 Contextualizing with CORGIS

The CORGIS project has been deployed for several semesters in introductory Computer Science courses for majors, ranging from the first course to a junior-level Data Structures course. These integrations ranged from small assignments to entire semester projects using the software. So far, the focus of the evaluation has been on the motivational influence of the system. Quantitative data was collected by surveying students attitudes using well-established motivational frameworks and instruments, and indicates that students tended to find real-time data engaging [4]. In some courses, qualitative data was gathered through small group interviews, where students attribute increased engagement with the authentic, real-world connection offered by real-time data.

Similar results have been found from its integration in a Computational Thinking course. Students cite working with the realistic data as a key factor in engaging with course materials. Refer to Figure 6, the result of an end-of-course survey (34 responses, 87% response rate). Students were asked to respond via a Likert scale to the statement, “Other schools teach

Computational Thinking in different ways. Consider these different styles, and indicate how much you would prefer or avoid them.” Although this is an extremely shallow question, since students were only exposed to two contexts (computational modelling and working with data), it is an encouraging, preliminary result.

5.4 BlockPy: Block-Based Programming with Python and Data Science

Programming is a difficult thing to learn, and there exist many tools to scaffold that process. One of the more popular approaches is block-based programming environments, which allow the user to build a program from pieces of provided code constructs. This prevents common syntax errors and some of the the “blank canvas” effect (the initial shock and disorientation a learner can suffer from when confronted with an empty slate). The CORGIS project has been used in several different block-based environments, including a Python-based system based on Blockly and the popular programming environment/language Snap!. Both systems implement their own layer around CORGIS in order to provide access to rich datasets. The BlockPy system has additional tools for creating visualizations of data, and the Snap! interface has generalized support for accessing online data [5]. The Snap! tools were created as part of a spin-off project by Jonathon Hellmann [28]. The BlockPy tools are my own work.

During the Computational Thinking course, students use the internal caching mechanism of the CORGIS libraries to work on recorded data. This simplifies the debugging process since programs perform predictably and are no longer dependent on external data services. When students are ready to run their programs against live data, they can move to a traditional programming environment (such as Spyder or PyCharm) and run in regular production. This caching mechanism also helps during assessment. A student’s program can be checked for robustness by invisibly changing the values returned by the big data functions. Figure 8 demonstrate the CORGIS library in action both with regular Python code and the equivalent Blockly code.

6 Intervention Context

With the development of new technology described in the following section, interventions will be staged through Virginia Tech’s new “Introduction to Computational Thinking” course, created to help fulfill the university’s new General Education Requirements [43]. This course has already been run for two semesters, deeply incorporating much of my existing research work. The course is taught and developed primarily by Dr. Dennis Kafura, although I have also been involved as associate instructor, managing course materials, server and technology administration, and assisting with in-class teaching. However, now that the course is more

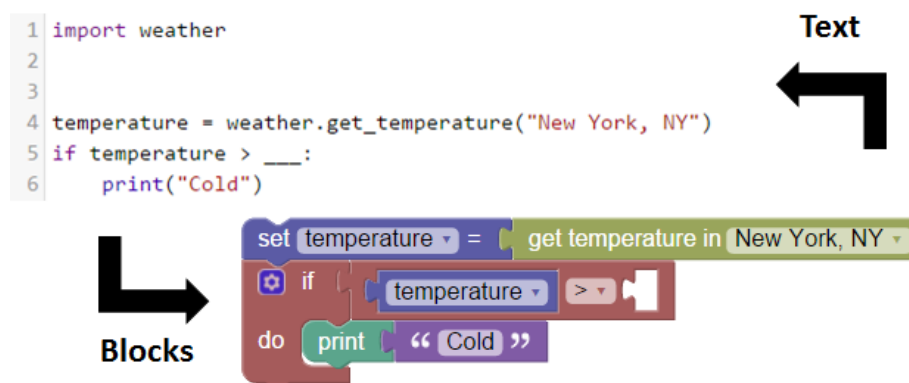


Figure 8: Blockly vs. Python Code

solidly defined, my role is shifting into a more observational function in order to drive my dissertation work. As a research endeavor, the course is heavily instrumented to provide data on its novel pedagogies and technologies. Although there are confounding factors to working with such a heavily experimental course, it presents a unique testbed for materials and is an excellent source for mining research results.

6.1 The Learners

The students in the Computational Thinking course present a unique profile. A few of them will have had prior programming experience, but most of them have had very minimal interactions with computers (indeed, they often describe themselves as “not a computer person”). These students may not believe that Computational Thinking will help them. This is largely because they have more clearly defined domain identities (that is, they have clearer career goals and established interests within their discipline), and may not see how Computational Thinking fits into them. So, indeed, these students often have low motivation, especially in their sense of Success and Usefulness.



In the first offering of the course, 25 students enrolled in the course, and 20 students finished the coursework. In the second offering, 40 students initially enrolled and 35 successfully passed the course. Figure 9 indicates the relevant demographic data collected through surveys. Largely, the students represent the population at Virginia Tech, albeit with some bias towards certain majors. It is worth pointing out the excellent gender diversity within the class.

Gender			Prior Programming Experience		
	Fall 2014	Spring 2015		Fall 2014	Spring 2015
Female	6	21	Yes	10	7
Male	13	18	No	10	32

Year			Colleges		
	Fall 2014	Spring 2015		Fall 2014	Spring 2015
Freshman	2	5	Engineering	2	2
Sophomore	7	11	Agriculture	0	1
Junior	6	11	Sciences	7	5
Senior	5	10	Liberal Arts	9	23
Unknown	0	2	Architecture	1	7
			Natural Resources	0	0

Figure 9: Demographic Data for Computational Thinking Students

6.2 The Content

Virginia Tech defines six learning objectives for “Computational and Quantitative Thinking” [43]. Although the Computational Thinking only satisfies four of these objectives outright, they are all considered valuable end-goals:

1. Explain the application of computational or quantitative thinking across multiple knowledge domains.
2. Apply the foundational principles of computational or quantitative thinking to frame a question and devise a solution in a particular field of study.
3. Identify the impacts of computing and information technology on humanity.
4. Construct a model based on computational methods to analyze complex or large-scale phenomenon.
5. Draw valid quantitative inferences about situations characterized by inherent uncertainty.
6. Evaluate conclusions drawn from or decisions based on quantitative data.

This content is mapped roughly into four instructional units on Computational Modelling, Algorithms, Data Intensive Inquiry, and Social Impacts. The Social Impacts unit is threaded throughout the course, while the other three are roughly sequential. Figure 10 gives a high-level overview of the content of this course.

It is clear that this material aligns smoothly with the content described in this preliminary proposal, in particular a focus on algorithms and abstraction.

Topic (Length)	Description
Computational Modeling(2 weeks)	Model-based investigation of how complex global behavior arises from the interaction of many “agents”, each operating according to local rules. Students use case-based reasoning and encounter basic computation constructs in a highly supportive simulation environment.
Fundamentals of Algorithms(4 weeks)	Study of the basic constructs of programming logic (sequence, decisions, and iteration) and program organization (procedures). A block-based programming language is used to avoid syntactic details. Students can see how these constructs are expressed in Python.
Data-intensive Inquiry(7 weeks)	Project-based exploration of complex phenomena by algorithmically manipulating large-scale data from real-world sources. Students construct algorithms in Python using a supportive framework for accessing the data.
Social Impacts(2 weeks)	Explore and discuss contemporary societal issues involving computing and information technology.

Figure 10: High-Level Course Overview

6.3 The Course

The course uses a considerable amount of modern pedagogical techniques, many of which represent ongoing research questions. Perhaps the most influential technique is the organization of students into cohorts. Near the beginning of the semester, students are put into groups of 5-6, balancing based on year and gender where possible, and avoiding putting similar majors into the same group. These cohorts primarily function as a support structure that students can rely on to get help and encouragement. Although cohorts work together on many smaller in-class assignments, every student is ultimately responsible for their own work. The final project, for instance, is individual to each student.

Class time is split between presentation (typically stand-up lecture) and participation (typically computer-based work or cohort discussion) using an Active Learning style wherever possible. Earlier assignments in the course often have students completing questions on paper or doing more kinetic exercises. Later assignments rely on the automated BlockPy questions, until the students reach the open-ended project work.

Work in the class is considered to employ a mastery style – students are allowed to attempt the material as many times as required. Deadlines are loose, so that students are free to work on the material as long as they need. A recurring message within the course is that “failing is okay, as long as you keep trying”.

7 Research Questions

In this section, I outline the problems that I will research. I’ve broken my proposed work into two main questions, each of which has a series of subquestions.

7.1 Research Question 1: Technology

What facilitations support the use of Data Science as an introductory context to provide motivation?

Data Science is a non-trivial context for introductory learners, due to the difficulty in finding, preparing, and delivering data to students in a pedagogically suitable form. Integrating it into a learning experience has already shown to be a tricky experience. In this subsection, I’ve outlined three research questions that I think are get at the heart of these challenges. To explore how to resolve these questions, I propose to build three new pieces of software:

CORGIS Architecture An evolution of the existing RTW architecture that handles new use cases and technical issues.

CORGIS Gallery An evolution of the existing RTW gallery that makes it easier for students and instructors to find relevant datasets.

CORGIS Builder An evolution of the existing RTW builder that makes it easier for developers to prepare data sources.

The remainder of this subsection motivates the research questions and explains how the proposed technology will help solve them.

7.1.1 RQ 1.1: CORGIS Library Architecture

How can we build and maintain a plethora of divergent datasets, that still ensure students have a uniform experience working with their dataset even in the presence of divergent hardware and datasets?

The primary value of the CORGIS project lies in the diversity of its datasets, giving students an empowered opportunity to find a dataset that appeals to their interests and long-term goals. The CORGIS library currently has over 35 different datasets, including animal feed data, weather reports, historical disease tracking, and much more. However, there is a large burden on the developer to create these datasets, requiring technical, pedagogical, and domain proficiency. And once these libraries are developed, they must be maintained: web-based libraries need to stay current with their API, and local libraries need to stay compliant with new hardware. Finally, these libraries have to be usable by students no matter what kind of hardware they have and whatever permissions they have on the machine.

Although the RealTimeWeb project greatly simplified the process of creating web-based libraries by using configuration files to generate libraries, this approach has failed to scale. Once a library has been generated, it becomes an independent code base with its own copy of the structure needed to access its data – essentially, we are inlining a tremendous amount of code. Worse, the novel features of the library become mired in boilerplate and library specializations. For example, consider the differences between the Gutenberg Books library and the Weather library, both of which expose a single function that connects to an online data source and returns a data structure mixing lists and maps: these two libraries are significantly similar except for their initial method to submit the web request and their final method that processes the retrieved data into the proper form. When an update is made to the web API, the developer must hunt down these two functions and make modifications, navigating a mess of boilerplate. Even worse is when improvements are needed to the core architecture. Despite sharing a common general architecture, each library is an independent code base with minute modifications. Therefore, a change to the architecture must be percolated to three dozen other codebases.

A second problem with the current architecture is the disorganization of the documentation and metadata that is associated with each library. Getting to know an API is a difficult process akin to learning to a new language. Supplementary documentation, including tutorials

and API references, are necessary. The RealTimeWeb project provided tools for documenting the libraries it generated, but these were limited to creating simple API reference materials that were not adaptable to different levels of learners and did not instruct the learner on its use; creating tutorials to use the libraries was a manual, cumbersome effort that was redundant across similar libraries. Further, RealTimeWeb had absolutely no tooling to generating supporting documentation related to metadata for the library – information such as origin of the data, explanation of terminology used, and terms of its use.

The third major problem is that students using our software can have different computational power: seniors might have a laptop from their freshman year, or run an older version of Mac OS. Ideally, the students should always be able to run their code quickly and efficiently while developing, without noticeable lag from their programs. However, several existing CORGIS libraries suffer greatly from bad caching strategies and poorly sized datasets, resulting in lousy performance that can frustrate beginners. A 100 MB library that runs fine on a developers new machine can be treacherously slow on a students’ ancient laptop.

RQ1.1 Solution: Create a new architecture that simplifies the creation of dataset libraries (whether web-based or local), while being highly maintainable to developers and performant for students.

Figure 12 shows my vision for the new architecture, highlighting the new components. In effect, all of the CORGIS libraries will be represented by one code base with “plug-and-play” data. These pluggable datasets will also incorporate structured rich metadata with an interface specification to indicate how students can access the data. Further, the library will be able to analyze the architectural suitability of the host machine and make intelligent decisions to adapt to the hardware. For example, if the software found that the students laptop had little RAM and a poor processor, it might decide to sample down the dataset, or to process more data on disk.

Once the solution is implemented, it will be evaluated based on case studies of creating new datasets and analyzing the work required to update existing datasets. The impact on the students’ experience will be analyzed through usability studies: students will be interviewed about their experience learning about the metadata and problems that they encountered while getting to know their dataset. More information on these usability studies is given in Research Question 2.1.

7.1.2 RQ 1.2: CORGIS Builder Architecture

How can we lower the barrier for instructors and domain experts to transform a data source into a classroom ready resource?

There is still too high a barrier for instructors to transform a data source into a classroom ready tool. Although our Real-Time Web Tool simplifies the process of connecting to web-based APIs, it has no features for simpler local datasets. Preparing a dataset is

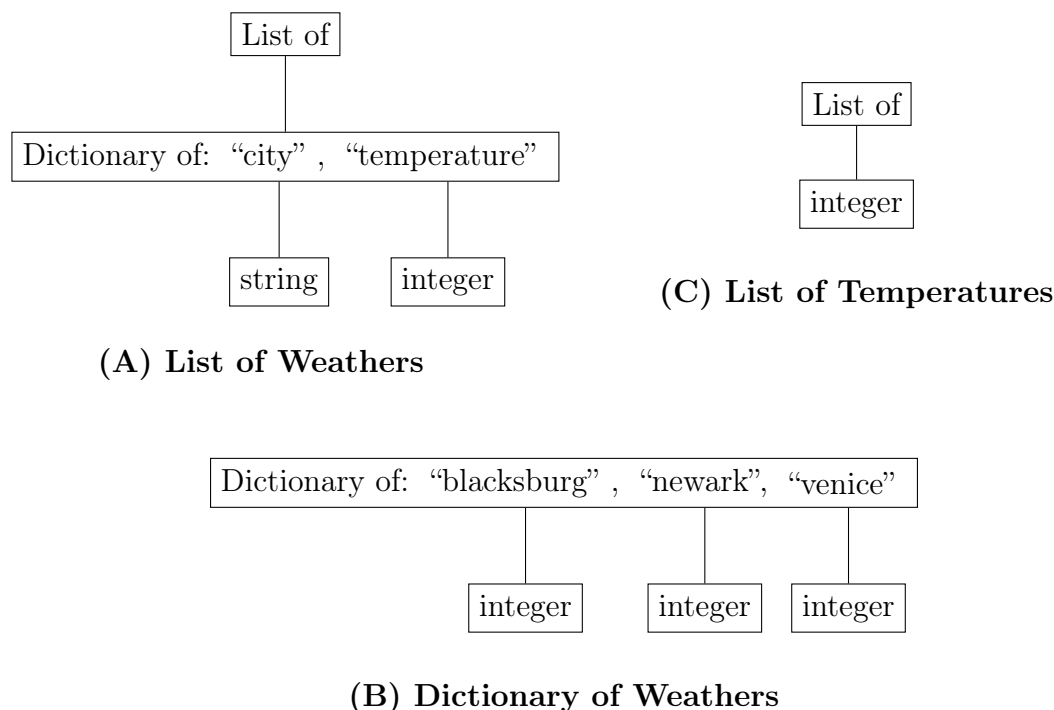


Figure 11: The same dataset can be structured differently according to the lesson at hand

an adhoc process of converting between data formats (e.g., JSON, CSV, SQL, etc.) into something manageable, requiring decisions about what fields and instances to keep, how the data should be structured hierarchically, what type fields should be, and how data should be pre-aggregated for students.

A further limitation is that the RTW Builder has no support for the process of building data caches. Instead, the instructor has to use the individual library to build up data caches using a poorly documented internal tool. This tool works in a cumbersome “VCR recording”-style, where the user runs the queries they’re interested in retaining in real-time. There is no way for the instructor to create artificial data caches matching their use case, without resorting to writing their own completely custom scripts.

Finally, different datasets have wildly varying structure depending on the nature of their data. A student working with social media data may find the data to be recursive or tree-structured, as opposed to a student with more tabular data working on sports statistics. Although this may be expected and natural, it is not desirable or necessary for students to have wildly divergent experiences with learning the structure of their data. If there is a uniform shape to the data, the instructor can have confidence and knowledge in providing technical and pedagogical support, no matter which student they are helping. Inversely, they can give a more uniform lecture that is accurate for all of the students.

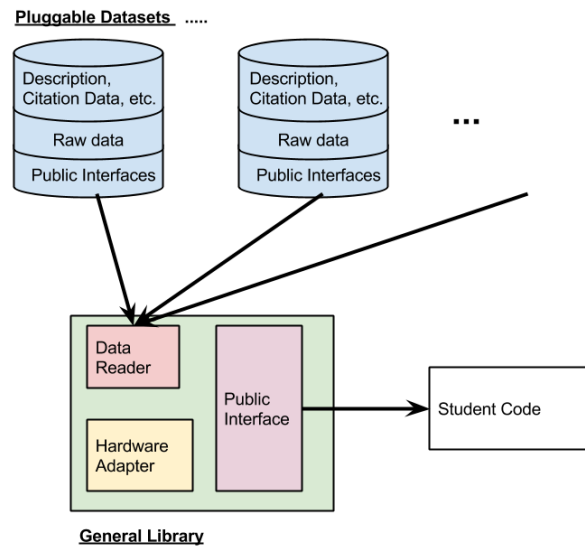


Figure 12: The proposed CORGIS Dataset Library Structure

RQ 1.2 Solution: An evolution of the RTW Online Building tool to make it even easier to prepare a JSON/CSV data source into a library.

This new version of the Online Building Tool will have features to reshape and organize a dataset, including ways to create data caches and artificial reworkings of the dataset according to instructor-supplied constraints. Specifically, the tool will be able to work with several different data formats, including CSV, JSON, SQL, and TXT, and be able to write definitions to connect to online APIs. Instructors will be able to write commands and queries to transform the data according to certain common functions or by using a regular query syntax. Figure 11 demonstrates how the same dataset can be parred down into different structures. Specifically, the tool will be able to manipulate datasets to have a desired shapes by restructuring the data according to certain common templates and high-level instructions given by the instructor. For example, consider a dataset of car makes and models over years with the columns ("year", "make", "model", "company"); this dataset could be grouped by year in order to make it easier for students to create bar charts in matplotlib (which would otherwise require the student to write grouping code). The other crucial new feature of the builder will be the ability to specify constraints and rules to generate artificial data for testing or to expand a data source, such as fake weather reports for the weather library. This use of mocking is a powerful way to provide more controlled learning experiences for learners. The output of the tool will be pluggable datasets suitable for the CORGIS architecture, rather than specific language bindings.

7.1.3 RQ 1.3: CORGIS Gallery

How can we support students' and instructors discovery of their data source?

Instructors have a choice to assign students a specific library, a choice of libraries, or to give students freedom to choose their own library. There are motivational and pedagogical trade-offs to consider, but this decision lies with the instructor. The goal of the CORGIS library is to allow the instructors to be as flexible as they want in assigning a dataset.

Currently, the list of CORGIS libraries is represented as a flat list of the libraries' names in a wiki structure ¹. Each library has an ad-hoc page of information which may or may not include code examples, library description, and a link to the source code. As the selection grows, this informal representation becomes more and more inadequate for finding a suitable library and learning more about its nature. Originally, the gallery was a dynamically generated page based on a separate specification of the libraries (separate from the libraries themselves) – this was too difficult to keep in sync with the libraries as they changed. The Wiki technology was adopted to make it simpler to make quick updates, but that just exacerbated the problem of keeping the library documentation up-to-date.

RQ 1.3 Solution: An enhancement to the RTW Online Gallery to make it more interactive and guiding for students to discover their datasets

I propose to make a new version of the RTW Gallery for the CORGIS project with three design goals:

1. Support instructors and students finding a suitable dataset. Provide features for both browsing and searching for libraries, especially for students who might have limited domain knowledge.
2. Support students looking up information about a library. Provide accessible information about the origin of the data source, the abstractions that it uses, citation data, information about the data's structure and fields, the interface exposed to access the data, any important limitations and features of the dataset, and other metadata relevant to the learner.
3. Keep the publicly available information of a dataset in sync with the datasets source. In particular, make it easy for developers to update the dataset or the metadata for the dataset, without requiring interaction with the server.

In the case of the first two design goals, success will be measured qualitatively through the motivation and usability interviews described in section 7.2.

¹<http://think.cs.vt.edu/wiki/index.php/Category:Library>

7.2 Research Question 2: Impact

How do the student-driven data context and scaffolds impact students' motivation and engagement?

In this proposal, I seek to study how the context and facilitations affect students self-reported motivation and the more quantifiable outcomes of engagement. The data to answer these questions will be collected over the course of a semester in an undergraduate Computational Thinking course. Students will be surveyed twice (pre-/post-) and interviewed individually about their experiences. Performance data will be collected in the course.

7.2.1 RQ 2.1: Motivation

What is the impact on motivation?

As previously described, the MUSIC Model states that Motivation comes through five different avenues: sense of empowerment (agency), sense of usefulness, sense of success (self-efficacy), interest, and a sense of being cared for. I seek to understand how a students' motivation to engage in the course changes over the semester, particularly in relation to the content, context, and scaffolds of the course.

RQ 2.1 Solution: Measured through self-reported quantitative surveys and qualitative interviews

The MUSIC model has an associated instrument named the Music Model of Academic Motivation Instrument (MMAMI) that can be used to quantitatively measure these attributes, but MMAMI will not be used in this study; the instrument is 26 questions long and is more targeted towards formative, holistic course revision than measuring the motivation attributable to course components. For instance, the instrument queries students about “the coursework”, and making the survey more specific extends the length too far. Instead, a custom survey based directly on the MUSIC model will be used to collect quantitative data about the students' motivation. This survey is given at the start of the semester (using the future-tense) and the end of the semester (using the past-tense).



The complete text of this survey is included in Appendix A. However, to summarize the survey, there are two main parts. The first half is five series of five 7-point likert statements (Strongly Disagree -> Strongly Agree). Each series relates to a different part of the MUSIC model (“I believe it will be interesting to...”), and each likert statement relates to a different part of the course:

- “... learn to write computer programs” - course content related to algorithms.
- “... learn to work with abstraction” - course content related to abstraction.

- "... learn about the social impacts of computing" - course content related to social ethics.
- "... work with real-world data related to my major" - course context related to data science.
- "... work with my cohort" - course scaffold related to the collaborative nature of the course.

These five elements were chosen as some of the most clear and important elements of the course that would be visible and understandable to a student at both the beginning and ending of the course. Although there are other course elements that would be desirable to gather data on (e.g., the block-based environment), there is a limited number of questions that we can ask students. These questions should be considered formative, as new elements might be added in future semesters. For now, these questions get at the three main course content objectives, the course context, and one of the most visible scaffolds available to students.

The second half of the survey instrument has five open-ended qualitative questions relating to the components of the MUSIC model, phrased to ask for particularly extreme examples of motivating and demotivating aspects: for example, the first question is "So far, what parts of the course seem particularly interesting or boring to you?". While the first half of the survey will give structured data about the components of the course, this section will give rough data about the "stand-out" parts. The data collected will be analyzed using a grounded method to establish recurring themes in the course components – for instance, a large percentage of students might report that the block-based environment made them feel particularly successful.

The data gathered using this instrument will be used to answer the following specific sub-questions:

1. What are students initial, self-reported attitudes and expectations entering the course?
2. How does students' motivation change over the course of the semester?
3. What course components are particularly effective or not effective at providing motivation?
4. Do different course components affect different aspects of motivation in different ways (e.g., scaffolds would be expected to affect success, but do they also have an impact on sense of usefulness or interest?)

In addition to the survey, a sample of students will be interviewed near the end of the semester in order to gather rich qualitative data specifically on the use of the data science

context (as opposed to the course components in general). I will attempt to select students to gather a representative distribution of the population as a whole, but this will be limited by the volunteer nature of the interviews. Questions will be asked relating to the students motivation towards the dataset, but also about the usability of their experience. The interview will be conducted over a period of roughly 30-45 minutes and will be audio recorded to make analysis easier.

The complete protocol for the interview is given in Appendix B. The interview will begin an introduction describing the overall goal, and some questions about the current status of the student. Then they will be questioned on their dataset in general and in relation to the five components of the music model (e.g., “What aspects of your dataset did you have control over?”). Finally, they will be asked an open-ended question about the dataset.

The data collected in the survey will be used to answer the following subquestions:

- What aspects of the dataset experience affected the students’ motivation?
- Do the students understand how to use the dataset?
- Do students generally expect the dataset to be authentic and relate to the real-world?
- How do students view the data science context in comparison to the rest of the course?

7.2.2 RQ 2.2: Engagement

What is the impact on engagement?

Within this proposal, I define engagement as the outcomes of motivation, as opposed to motivation itself which is internal and non-observable. There are a wide range of outcomes that can be classified as potential outcomes of engagement. These outcomes represent desirable behavior from students that should be affected by their internal motivation. I will analyze students engagement in order to find what can be predicted by their motivation.



RQ 2.1 Solution: Measured through observable outcomes and some self-reported information

In order to do this analysis, I will collect several different outcomes of their observed behavior. The following is a list of the outcomes I will be measuring, and their source:

Attendance Measured as a numeric value indicating how many classes they attended. This is a normal part of the course metrics, since attendance influences a students’ grade.

Procrastination Measured as a number indicating how many times the student handed in an assignment late.

Assistant Observations Measured as two numbers that reflect the students' primary teaching assistant assessment of the students' motivation and ability. This will be collected near the end of the semester.

Course Performance Measured as several numbers based on the students' performance in each of the different phases of the course.

Project Performance Measured as a cumulative number based on the students' performance on the final data science project (which is evaluated according to a rubric by multiple reviewers within the course).

Intent to Continue The survey includes a single engagement question, asking if students intend to continue their computing education (7-point likert, Strongly Disagree to Strongly Agree).

It is possible that further self-reported outcomes could be added to this list in the future.

I will analyze the relationship of each of these outcomes with the motivation data collected in the first question. In particular, I am looking for what aspects of motivation can be used to predict engagement outcomes. For example, preliminary data (shown in Figure 13) gathered in prior versions of the survey suggests that students' sense of the usefulness of the material is a more accurate predictor of whether they intend to continue in computing than their interest in the material (N=35). The data collected in that figure comes from a survey more strongly based off MMAMI, which means that it only asks about motivation at the course level, not at the course component level. With the new survey instrument, I hope to find out whether student's motivation towards some course elements are better predictors of the engagement outcomes than others. I also expect to see improvements in students' engagement outcomes related to the data science project as the technology backing the project (proposed in the previous subsection) is improved from semester to semester.



8 Work and Publication Plan

This section outlines a schedule for my research over the next two years, starting from this fall.

- Fall 2015
 - SIGCSE'16 Paper describing Python CORGIS integration into Blockly, the block-based programming environment used in the course (submitted)
 - Third offering of CT course, collect data (N=35)
 - First version of new CORGIS Architecture

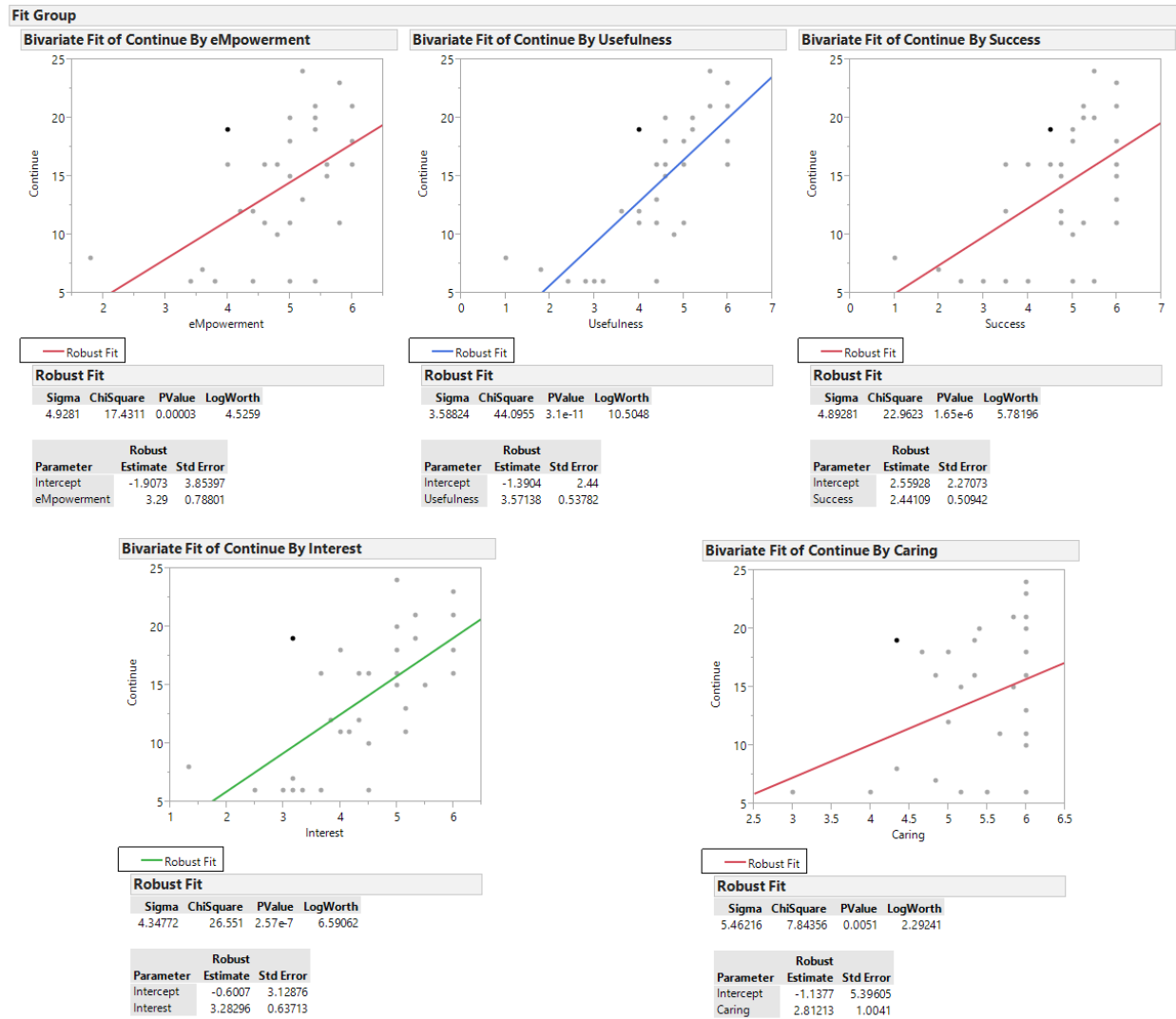


Figure 13: Students intent to continue their computing education vs. their self-reported motivation. Usefulness might be a better predictor than Interest or Caring (N=35).

- Spring 2016
 - Fourth offering of CT course, collect data (N=30-60?)
 - First version of the new CORGIS Gallery
- Summer 2016
 - First version of the new CORGIS Builder
- Fall 2016
 - Finalization of all major CORGIS architectures
 - Fifth offering of CT course, collect data (N= expected 80?)
 - SIGCSE'17 Paper on the CORGIS project
- Spring 2017
 - Sixth offering of CT course, collect data (N=80?)
 - TOCE Journal Paper as a summative view of my primary research questions (using CORGIS to meaningfully motivate and guide large quantities of students to success).
 - Final Thesis Defense

9 Conclusion

My research is focused on motivating introductory computer science classes with broadly applicable contexts. I seek to introduce new tools and approaches that can support instructors, developers, and learners in using data science and datasets as an introductory context. In this preliminary proposal, I describe the technical work and how it will be deployed and evaluated in real classroom settings. I anticipate that this approach to introductory computing will provide useful new techniques and tools instructors. The research results that I uncover will advance the literature. Thank you for taking the time to read through it!

References

- [1] R. E. Anderson, M. D. Ernst, R. Ordóñez, P. Pham, and B. Tribelhorn. A data programming cs1 course. In *Proceedings of the 46th ACM Technical Symposium on Computer*

- Science Education*, SIGCSE '15, pages 150–155, New York, NY, USA, 2015. ACM.
- [2] R. E. Anderson, M. D. Ernst, R. Ordóñez, P. Pham, and S. A. Wolfman. Introductory programming meets the real world: Using real problems and data in cs1. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 465–466, New York, NY, USA, 2014. ACM.
 - [3] L. J. Barker, C. McDowell, and K. Kalahar. Exploring factors that influence computer science introductory course students to persist in the major. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, SIGCSE '09, pages 153–157, New York, NY, USA, 2009. ACM.
 - [4] A. C. Bart, E. Tilevich, S. Hall, T. Allevato, and C. A. Shaffer. Transforming introductory computer science projects via real-time web data. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 289–294, New York, NY, USA, 2014. ACM.
 - [5] A. C. Bart, E. Tilevich, C. Shaffer, and D. Kafura. Position paper: From interest to usefulness with blockpy, a block-based, educational environment. *Blocks and Beyond '15*, October 2015.
 - [6] M. Ben-Ari. Situated learning in computer science education. *Computer Science Education*, 14(2):85–100, 2004.
 - [7] J. S. Brown, A. Collins, and P. Duguid. Situated cognition and the culture of learning. *Educational researcher*, 18(1):32–42, 1989.
 - [8] M. Buckley, J. Nordlinger, and D. Subramanian. Socially relevant computing. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, pages 347–351, New York, NY, USA, 2008. ACM.
 - [9] A. Bundy. Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, 1(2):67–69, 2007.
 - [10] J. Carter, D. Bouvier, R. Cardell-Oliver, M. Hamilton, S. Kurkovsky, S. Markham, O. W. McClung, R. McDermott, C. Riedesel, J. Shi, and S. White. Motivating all our students? In *Proceedings of the 16th Annual Conference Reports on Innovation and Technology in Computer Science Education - Working Group Reports*, ITiCSE-WGR '11, pages 1–18, New York, NY, USA, 2011. ACM.
 - [11] L. N. Cassel. Interdisciplinary computing is the answer: Now, what was the question? *ACM Inroads*, 2(1):4–6, 2011.
 - [12] J.-I. Choi and M. Hannafin. Situated cognition and learning environments: Roles, structures, and implications for design. *Educational Technology Research and Development*, 43(2):53–69, 1995.

- [13] W. Commons. Data science process, 2014.
- [14] C. Day. Computational thinking is becoming one of the three R's. *Computing in Science and Engineering*, 13(1):88–88, 2011.
- [15] P. J. Denning. The profession of IT: Beyond computational thinking. *Communications of the ACM*, 52(6):28–30, 2009.
- [16] P. J. Denning and A. McGettrick. Recentering computer science. *Commun. ACM*, 48(11):15–19, #nov# 2005.
- [17] P. DePasquale. Exploiting on-line data sources as the basis of programming projects. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '06, pages 283–287, New York, NY, USA, 2006. ACM.
- [18] B. DiSalvo and A. Bruckman. From interests to values. *Commun. ACM*, 54(8):27–29, Aug. 2011.
- [19] L. Douglas. The importance of ‘big data’: A definition. *Gartner (June 2012)*, 2012.
- [20] A. E. Egger. Engaging students in earthquakes via real-time data and decisions. *Science*, 336(6089):1654–1655, 2012.
- [21] A. Erkan, T. Pfaff, J. Hamilton, and M. Rogers. Sustainability themed problem solving in data structures and algorithms. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, SIGCSE '12, pages 9–14, New York, NY, USA, 2012. ACM.
- [22] I. S. for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA). Operational definition of computational thinking, 2011.
- [23] A. Forte and M. Guzdial. Computers for communication, not calculation: Media as a motivation and context for learning. In *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 4 - Volume 4*, HICSS '04, pages 40096.1–, Washington, DC, USA, 2004. IEEE Computer Society.
- [24] M. Goldweber, J. Barr, T. Clear, R. Davoli, S. Mann, E. Patitsas, and S. Portnoff. A framework for enhancing the social good in computing education: A values approach. In *Proceedings of the Final Reports on Innovation and Technology in Computer Science Education 2012 Working Groups*, ITiCSE-WGR '12, pages 16–38, New York, NY, USA, 2012. ACM.
- [25] M. Guzdial and A. E. Tew. Imagineering inauthentic legitimate peripheral participation: An instructional design approach for motivating computing education. In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 51–58, New York, NY, USA, 2006. ACM.

- [26] O. A. Hall-Holt and K. R. Sanft. Statistics-infused introduction to computer science. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 138–143, New York, NY, USA, 2015. ACM.
- [27] S. F. Harp and R. E. Mayer. How seductive details do their damage: A theory of cognitive interest in science learning. *Journal of Educational Psychology*, 90(3):414, 1998.
- [28] J. D. Hellmann. *DataSnap: Enabling Domain Experts and Introductory Programmers to Process Big Data in a Block-Based Programming Language*. PhD thesis, Virginia Tech, 2015.
- [29] D. Hemmendinger. A plea for modesty. *ACM Inroads*, 1(2):4–7, 2010.
- [30] N. Iarocci and G. Amica. Eve. <https://github.com/nicolaiarocci/eve>, 2012.
- [31] G. Inc. Exploring computational thinking, 2011.
- [32] J. A. Jacobs. Interdisciplinary hype, November 2009.
- [33] B. D. Jones. Motivating students to engage in learning: The MUSIC model of academic motivation. *International Journal of Teaching and Learning in Higher Education*, 21(2):272–285, 2009.
- [34] B. D. Jones and G. Skaggs. *Validation of the MUSIC Model of Academic Motivation Inventory: A measure of students' motivation in college courses*. Research presented at the International Conference on Motivation 2012, 2012.
- [35] J. S. Kay. Contextualized approaches to introductory computer science: The key to making computer science relevant or simply bait and switch? In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 177–182, New York, NY, USA, 2011. ACM.
- [36] J. Kramer. Is abstraction the key to computing? *Commun. ACM*, 50(4):36–42, Apr. 2007.
- [37] J. Lave and E. Wenger. *Situated learning: Legitimate peripheral participation*. Cambridge university press, 1991.
- [38] L. Layman, L. Williams, and K. Slaten. Note to self: Make assignments meaningful. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 459–463, New York, NY, USA, 2007. ACM.
- [39] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers. Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*, 2011.

- [40] J. J. McGinley and B. D. Jones. A brief instructional intervention to increase students' motivation on the first day of class. *Teaching of Psychology*, 41(2):158–162, 2014.
- [41] M. Mitchell, J. Sheard, and S. Markham. Student motivation and positive impressions of computing subjects. In *Proceedings of the Australasian Conference on Computing Education*, ACSE '00, pages 189–194, New York, NY, USA, 2000. ACM.
- [42] S. Mneimneh. Fibonacci in the curriculum: Not just a bad recurrence. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 253–258, New York, NY, USA, 2015. ACM.
- [43] O. of the Senior Vice President and Provost. Academic implementation strategy for a plan for a new horizon: Envisioning virginia tech 2013-2018. Technical report, Virginia Tech, 2013.
- [44] O. L. Oliveira, A. M. Monteiro, and N. T. Roman. From concrete to abstract?: Problem domain in the learning of introductory programming. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, pages 173–177, New York, NY, USA, 2011. ACM.
- [45] A.-C. J. T. F. on Computing Curricula. Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press, December 2013.
- [46] S. Papert. An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1):95–123, 1996.
- [47] J. Petraglia. The real world on a short leash: The (mis) application of constructivism to the design of educational technology. *Educational Technology Research and Development*, 46(3):53–65, 1998.
- [48] J. Pfeiffer. *The thinking machine*. Philadelphia, PA: Lippincott, 1962.
- [49] C. Rader, D. Hakkarinen, B. M. Moskal, and K. Hellman. Exploring the appeal of socially relevant computing: Are students interested in socially relevant problems? In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 423–428, New York, NY, USA, 2011. ACM.
- [50] B. E. Rogoff and J. E. Lave. *Everyday cognition: Its development in social context*. Harvard University Press, 1984.
- [51] Y. N. Silva, S. W. Dietrich, J. M. Reed, and L. M. Tsosie. Integrating big data into the computing curricula. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 139–144, New York, NY, USA, 2014. ACM.
- [52] D. E. Stevenson and P. J. Wagner. Developing real-world programming assignments for cs1. *SIGCSE Bull.*, 38(3):158–162, June 2006.

- [53] J. A. Stone and E. M. Madigan. The impact of providing project choices in cs1. *SIGCSE Bull.*, 40(2):65–68, June 2008.
- [54] D. G. Sullivan. A data-centric introduction to computer science for non-majors. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 71–76, New York, NY, USA, 2013. ACM.
- [55] T. Thompson and R. Barnes. The engine of successful education reform: Effective teachers and principals. *Commission on No Child Left Behind. The Aspen Institute. Denver, (Oct. 2009)*, 2009.
- [56] L. Torgo. *Data Mining with R, learning with case studies*. Chapman and Hall/CRC, 2010.
- [57] M. Waldman. Keeping it real: utilizing NYC open data in an introduction to database systems course. *J. Comput. Sci. Coll.*, 28(6):156–161, #jun# 2013.
- [58] A. Weinberg. Computational thinking: An investigation of the existing scholarship and research. In *School of Education*. Fort Collins, Colorado State University, 2013.
- [59] Z. Weiner. How introductory physics problems are written. <http://www.smbc-comics.com/comics/20131202.png>, Dec 2013.
- [60] G. Wilson. Software carpentry: Getting scientists to write better code by making them more productive. *Computing in Science and Engg.*, 8(6):66–69, Nov. 2006.
- [61] J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.
- [62] S. Yarosh and M. Guzdial. Narrating data structures: The role of context in cs2. *Journal on Educational Resources in Computing (JERIC)*, 7(4):6, 2008.
- [63] Z. Zografski. Innovating introductory computer science courses: Approaches and comparisons. In *Proceedings of the 45th Annual Southeast Regional Conference*, ACM-SE 45, pages 478–483, New York, NY, USA, 2007. ACM.

A Motivation Survey (Pre)



Your participation is voluntary, but we value your opinion and will use your responses to make changes to the course in the future. We would like a high response rate to ensure that our findings are representative of everyone in the course. If you do not wish to participate, simply do not fill out the survey. You must be 18 or older to take part in this research.

Responses will not affect your grade at all - they are only used for research and course improvement purposes.

I believe that it will be interesting to...

[illegible]

I believe that it will be useful to my long-term career goals to...

[illegible]

[illegible]

I believe that I will try to learn more about computing (e.g., take an online course, work on a personal computing project, etc.) on my own after this course.

Strongly Disagree Disagree Somewhat Disagree Neither Agree nor Disagree Somewhat Agree Agree Strongly Agree

☐ ☐ ☐ ☐ ☐ ☐ ☐

The following questions are open-ended. Please answer them as briefly or as in-depth as you'd like.


So far, what parts of the course seem particularly interesting or boring to you?


So far, what parts of the course seem particularly useful or useless to your long-term career goals?


So far, what parts of the course seem particularly hard or easy to you?

So far, what parts of the course seem particularly empowering or limiting to you?



So far, how have the **instructors or your classmates** shown that they care or don't care **about you?** 





B Motivation Interview

Attachment G - Sample Interview Protocol (Fall 2015)

This is a sample Interview protocol for the study on students' perceptions of the components of the course and their impact on the students' motivation.

Take permission to audio record the session, explain that participants will remain anonymous.

Introduction: I am interested in understanding how people experience this course. For our time together, I'd like to discuss how engaged you were in the course and what parts motivated and demotivated you, specifically in the CS 2984 - Introduction to Computational Thinking class. If at any time you feel uncomfortable with a question or where our discussion is going, just let me know. We can stop at any time. Do you have any questions before we get started?

- **Can you tell me a bit about your background?**
 - Which program are you enrolled in?
 - What other courses were you taking this semester?
 - Why are you taking this Computational Thinking course?
- **Can you explain a typical computational thinking class?**
 - How was the class going for you?
 - Are there things about the class that are helpful?
 - Are there things about the class that you wish were different?
- **Tell me about the dataset you're using for the Computational Thinking class.**
 - What was its name?
 - What kind of data did it have?
- **What aspects of your dataset did you have control over?**
 - What did you feel limited by with your dataset?
 - How did you decide to choose to your dataset?
 - Did you want to know more about how it worked internally?
 - How does this compare to the rest of the course? Was it more or less controlled?
- **How well does your dataset align to your career goals?**
 - What problems were you able to solve with your dataset?

- What couldn't you do with your dataset?
- How does this compare to the rest of the course?
- **How did you get to know your dataset?**
 - What things still confuse you about your dataset?
 - When you opened your dataset, did you know what to do?
 - How does this compare to the rest of the course?
- **Was your dataset interesting?**
 - What was boring about your dataset?
 - How does this compare to the rest of the course?
- **Do you feel that this dataset would be generally valued by your peers in your program?**
 - Do you feel that this dataset would be generally valued by your peers in this class?
 - Do you feel that this dataset would be generally valued by the instructors?
 - How does this compare to the rest of the course?
- **Based on our conversation, is there anything else you want to tell me about using your dataset?**

Thank you for your participation!