Computing with CORGIS: Diverse, Real-world Datasets for Introductory Computing

Austin Cory Bart, Ryan Whitcomb, Eli Tilevich, Dennis Kafura, and Clifford A. Shaffer
{acbart, rwhit94, tilevich, kafura, shaffer}@vt.edu
Virginia Tech

ABSTRACT

Bringing introductory computing to non-CS majors means creating a curriculum that will appeal to students from diverse disciplines. Many educational theories emphasize the need for introductory contexts that are perceived as useful and align with students' long-term goals. Data Science, using algorithms to manipulate real-world data and interpreting the results, has emerged as a field with cross-disciplinary value, and has strong potential as an appealing context for introductory computing courses. However, it is not easy to find, clean, and integrate datasets that will satisfy a broad variety of learners. The CORGIS project makes it easier for instructors to incorporate data science into their classroom. We provide over 50 datasets [UPDATE FINAL NUMBER BEFORE RELEASE], including topics from history, politics, medicine, and education. The CORGIS infrastructure makes it easy to integrate a new dataset with simple libraries for Java, Python, and Racket, so that introductory students can write programs that manipulate real data. Web-based tools also allow learners to visualize and explore datasets without programming, enabling data science lessons on day one. We describe the mechanics of the system and our experience using the libraries, and share research results on Real-world Data in an introductory context.

CCS Concepts

•**Information systems** → **Information integration;** Data management systems; •**Social and professional topics** → *Computational thinking; Model curricula;* Computing literacy;

Keywords

corgis; data; data science; pedagogy; motivation; microworld; real-world; authentic; big data

SIGCSE '17 Seattle, Washington USA

DOI:

1. INTRODUCTION

As computing skills are becoming required for an increasing number of disciplines, CS educators are meeting the demand by creating courses for non-CS majors (e.g., "Computational Thinking"). Students are drawn from many disciplines, including the sciences, arts, and humanities, and are distinctive from Computer Science majors in that they typically have many different career paths before them. Some curricula for non-CS majors emphasize game and media design as a way to draw these students into computing, but these topics do not reflect how most non-majors will use computing in their jobs – a fact that learners are aware of [9].

We suggest that most students will eventually need to draw on computational techniques in order to manage the ever-growing waves of data present in every field [14]. Therefore, a Data Science context, where students write algorithms to manipulate real-world data and interpret the results, could provide a compelling context. However, integrating data into introductory courses creates challenges for instructors, both pedagogically and technologically. To support this approach, we present here the open-source CORGIS project, the **C**ollection **O**f **R**eally **G**reat and **I**nteresting data**S**ets. Our materials are free and open-source, available at https://think.cs.vt.edu/corgis.

This paper begins with a brief review of relevant educational theory and existing projects in this space. We then describe the technical innovations of the CORGIS project and its pedagogical affordances. We report and evaluate results of an intervention conducted using the software. Finally, we discuss the future of the project.

2. EDUCATIONAL THEORY

Educational Theory grounds the development of the CORGIS project. We use Situated Learning Theory [13] to better understand how students learn, and the MUSIC Model of Academic Motivation [12] to better understand why students choose to engage.

Situated Learning Theory suggests that authentic contexts are crucial for learners. Originally proposed by Lave and Wenger, SL Theory argues that tasks in the learning environment should parallel real-world tasks, in order to maximize the authenticity of the experience [13]. Some interpretations of this theory draw a distinction between the content (e.g., learning to program) and the context (e.g., by creating a video game), and stress that proper contextualization is important for students' comprehension and investment [5].

To understand motivation, we rely on the the MUSIC Model of Academic Motivation, a holistic model of moti-

vation [12]. Built as a meta-model, it incorporates many existing theories and is tailored particularly for education. The model distinguishes between students' situational *interest* in an academic activity and their sense of the *usefulness* of the activity to their long-term career goals. We connect this distinction to the different contexts available to introductory courses. Creating games and animations appeals to students' situational interest, while we posit that data science will appeal both to students' situational interest and their sense of usefulness. The MUSIC model also incorporates students' expectation for appropriate levels of *Success* when engaging in activities, their sense of *eMpowerment* and agency within the activity, and their perception of their instructors and classmates *Care* towards them.

Supported by these two theories, we hypothesize that Data Science will be a beneficial context for students, since we believe that they will perceive it as closely related to their long-term career goals.

3. RELATED AND PRIOR WORK

The use of data analysis for contextualization is not novel, and represents a new and actively growing movement [1, 16, 10, 6]. Upper division courses have employed situated learning experiences using data of varying size and complexity for several years [7, 17]. However, this prior work has little evaluation of the advantages and disadvantages of data science in introductory education [16, 6].

The RealTimeWeb project [2] made it easy for instructors to bring web-based, constantly-updating data into their introductory classrooms by writing light-weight specifications of remote data streams. CORGIS is a direct successor to the RTW project, moving from a small collection of dynamic datasets to a wide collection of static datasets. The change was motivated by the scarcity of quality, real-time datasets. And while the technical challenges of integrating real-time data were interesting, they were of secondary importance to the value of having a diversity of relevant datasets.

The BigDataCSE project of Hamid [11] takes real-time data access in a different direction by reducing the technical requirements on instructors even further. The library uses sophisticated reflection techniques to automatically infer the structure of data, so that students can access any desired URL endpoint and receive structured data. Although it provides a flexible architecture, this project does not attempt to provide datasets, making it more appropriate for advanced students who can find their own data to work with.

The BRIDGES project provides students with visualizations of their algorithms on datasets [4]. BRIDGES does not focus on organizing datasets directly, but incorporates existing datasets (including some directly from the CORGIS project). The use of visualizations can help students understand the interaction between their algorithms and data.

4. DESIGN

Figure 1 provides an overview of how datasets are added to the CORGIS project. First, real-world data is collected and preprocessed into a "clean" state. Next, high-level specification files are written to describe the data. These specification files are used in the final phase by our compiler to generate programming language bindings, data files, and other output targets.

4.1 Finding Datasets

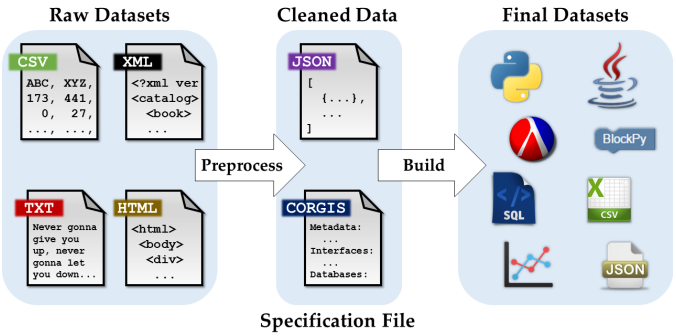

Figure 1: How datasets are added to CORGIS

The CORGIS project aspires to have multiple (3-4) datasets relevant to each major career path that potential students might seek. To achieve this, we draw on data from wide-ranging, open-access sources including governments, research publications, journalists, non-profits, and industry. We only use datasets that are publicly licensed, so that students do not have to sign non-disclosure agreements. Unfortunately, this limits access to many excellent datasets.

4.2 Dataset Preparation

Every dataset requires some ad-hoc cleaning. Most of the human effort for the project comes at this phase, as organizing a dataset requires expertise. The variety of the formats and shapes of the data used prevent a one-size-fits-all methodology. In general, we attempt to formalize the process by maintaining a "raw" dataset, a preprocessing script, and the resulting "cleaned" dataset. Most of the existing scripts are written in Python, using popular data manipulation libraries such as Pandas. Examples of our methodology can be found on the CORGIS Github repository.

Figure 5 is a representation of the cleaned structure of the Classic Books dataset. Every dataset is a list of hierarchical maps (a tree), where the leaf nodes are simple data types: integers, real numbers, text, and booleans. Ideally, the dataset should fit comfortably into memory, so we target around 5-10MB worth of data. The structure of the data creates opportunities for lessons on collection-based iteration and nested data structure accessing.

4.3 Specification Files

CORGIS specification files use a YAML-based format that defines metadata and interfaces for a cleaned data file. Specification are compiled and validated according to a schema. Optional comments for individual fields can be included, for example to indicate units or peculiarities of the data. All datasets in CORGIS have interfaces that are implemented as SQL queries that can be used to query the dataset. Interfaces provide access points into the data for students. Every dataset comes with a default interface that simply returns the entire dataset with no modification. Many datasets provide interfaces to access individual elements or slices across elements. The weather dataset, for example, provides "Get Temperature" and "Get Past Temperatures" interfaces that return an integer and a list of integers, respectively.

4.4 Final Datasets

```python
# Python
import crime
crimes = crime.get_all()
```

```
; Racket
(require crime)
(define reports (crime-get-all))
```

```java
// Java
import corgis.crime.StateCrimeLibrary;
import corgis.crime.domain.Report;

public class Main {
  public static void main(String[] args) {
    StateCrimeLibrary stateCrimeLibrary =
        new StateCrimeLibrary();
    ArrayList<Report> reports =
        stateCrimeLibrary.getAllCrimes();
  }
}
```

Figure 2: Example of using a CORGIS library

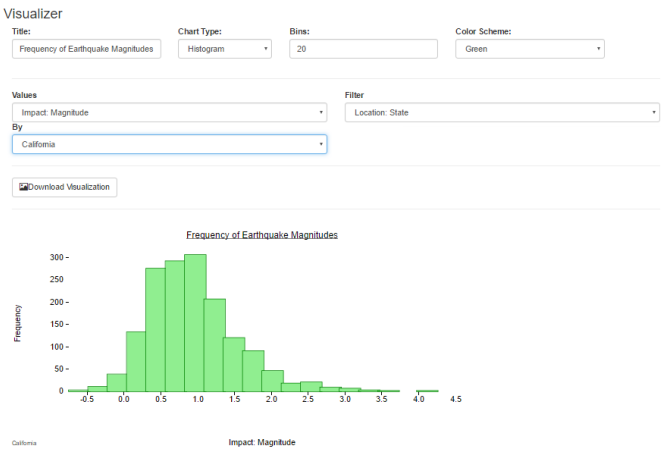

Figure 3: The CORGIS visualizer

Given a specification file and the associated dataset, the CORGIS builder can generate client libraries for various languages, well-structured raw data files, and an interactive interface for experimenting with data in the browser.

4.4.1 Language Libraries

The CORGIS project currently supports Python, Java, and Racket. Figure 2 gives examples of how a generated library can be used in these languages. The Python version of the CORGIS library uses only native Python types: lists, dictionaries, ints, floats, strings, and booleans. It is compatible with Python 2.7, Python 3.5, and BlockPy [3]. The Racket version is similar to the Python version, both in implementation and interface. The Java version uses custom classes instead of dictionaries.

Code and supporting documentation for a library is generated by filling out templates using the Jinja2 templating library. Interfaces from the specification file generate functions that execute queries. A SQLite database is also generated to store the data, rather than the original JSON file, for performance reasons. However, the database stores each top-level element of a dataset as a JSON-encoded string, along with any indexed fields. The benefit of this key-value store structure is that the dataset can be efficiently sampled on-disk while still retaining a hierarchical structure that normally would not fit nicely into a relational database.

4.4.2 Visualizer

The Visualizer allows students to engage with datasets in their browser without programming, as shown in Figure 3. Users can generate histograms, line plots, scatter plots, and bar charts for any supported dataset. Bar charts are categorized by the indexes from the specification file. Datasets can also be filtered on indexed fields when generating charts. Although not as powerful as a full programming environment, the Visualizer empowers students to work with real-world data without specialized training.

4.4.3 Explorer

Separate from the data content, students can explore the structure of a dataset using the Explorer. Emulating a similar feature in the Spyder IDE[1], the hierarchy of data is explored through chained windows. Clicking a field in a map, for instance, might open a new map's window, which in turn might have atomic fields or links to further maps.

[1]https://pythonhosted.org/spyder/

4.4.4 Raw Data Files

Language bindings simplify access to datasets, but are not realistic representations of how to access real-world data. Therefore, we also expose raw datasets in popular data formats: SQL, CSV, and JSON. The JSON format is a straight dump of the data file. The SQL database represents each child map as a distinct table, linked with a primary key. The CSV table is a flattened representation of the data, using the key hierarchy to disambiguate column names.

4.5 The CORGIS Gallery

Language bindings, data files, and the Visualizer are all accessible through a user-friendly Gallery, shown in Figure 4. Every dataset is documented with a list of content tags to help students search for a relevant dataset.

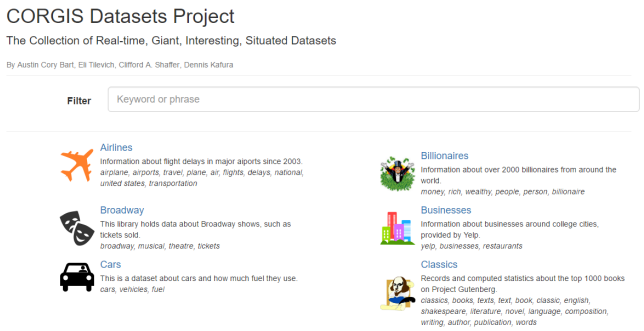

Figure 4: The CORGIS gallery

5. ASSIGNMENTS

CORGIS can be integrated into introductory courses in a number of ways. In this section, we describe how we have used the tools in an existing course on Computational Thinking for non-majors. This course teaches basic programming in a Data Science context. Much of the CORGIS development has been driven by the needs of the course.

5.1 Exploratory Analysis

The Visualizer facilitates early exploration of datasets, without the need to program. In the beginning of our course, students generate graphs in groups (each group is assigned a dataset) and then on their own (free to choose their own dataset). This gives students practice selecting and interpreting different kinds of charts. In our experience, many students struggle with aspects of graphs such as distinguishing bar charts and histograms, or knowing when to use line plots.

5.2 Practice Problems

When students start programming, we give them practice problems contextualized with CORGIS datasets. Although the scope of these problems is similar to those found in systems like CodingBat [15], the problems can be more realistic. The complexity of using complete datasets is avoided by using the simpler interfaces exposed for libraries. For example, students might be tasked with writing a program to print whether an umbrella is necessary depending on the weather in their current city (requiring only a function call to the Weather library, an if-statement, and the print statement).

5.3 Data Mapping

In order to process data, students must first understand its structure. We direct students to create a hand-written "Data Map", like the one in Figure 5. This diagram visually describes the structure of the dataset, and students can use it as a guide to developing the necessary code constructs (e.g., iteration, dictionary access), much as they would navigate a real map to find a path in a maze. Because of the vastness of most datasets' structure, students must be judicious in what branches and leaves of the structure they diagram, in anticipation of what they will use in their code.

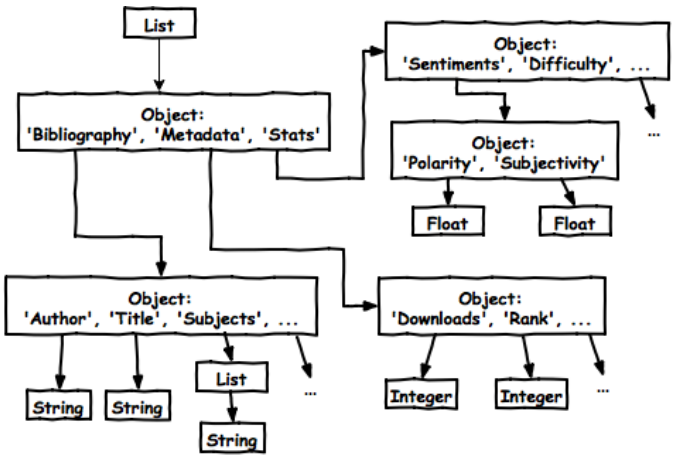

Figure 5: Partial data map for the Classics library

5.4 Data Analysis

For our course final project, students are tasked with formulating questions about a dataset of their own choosing, writing code to create visualizations, and then interpreting those visualizations using relevant domain knowledge. We claim that this is an authentic form of assessment for students in the course, modeling what they might do if they were to incorporate data science into their own careers. In addition to their code, students must turn in a 5-minute video presentation reporting their results. Students share their videos with each other to demonstrate the breadth available in computing. Students are required to describe the abstractions used in the project and the inherent limitations of the dataset. This turns a weakness in the dataset into an important learning experience for the students. The final project is assessed both by course staff and peers.

6. EVALUATION

In this section, we evaluate the projects' progress in two ways. First, we present empirical metrics for the datasets. Second, we present survey results from a course that incorporated real-world data through CORGIS.

6.1 Metrics

At the time of writing, there are 43 datasets in the CORGIS gallery, and we are actively working to expand. The left graph of Figure 6 reveals characteristics of the datasets within the corpus. If the data's structure is seen as a tree, the Average Branch Factor (ABF) is the mean number of fields in a child. The Height of a dataset is the maximal depth of the tree, and Fields is the number of leaves. Rows (measured in thousands) is the number of records in the dataset, while Size is the amount of disk space used by a dataset. Although we are pleased with the narrow distribution on some of the attributes (e.g., heights), the dispersion of ABF and Fields suggests that some datasets need to have fewer fields organized into more branches.

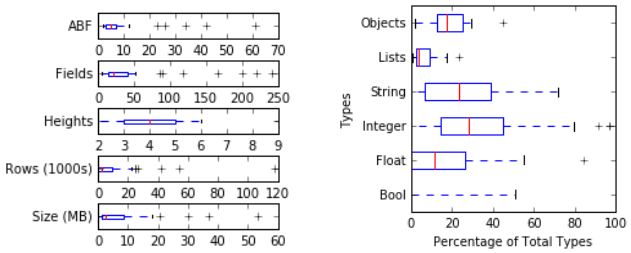

Figure 6: Distribution of structure, characteristics, and types across all datasets

The right graph of Figure 6 shows the distribution of atomic and composite types within datasets. Numeric and string types dominate. Most datasets have few or no boolean types, and few datasets have more than just the top-level list (which is present in every dataset). The x-axis shows percentages of all types within the dataset, with numerics and strings as the most common. The chart does not distinguish between different types of string values, such as unique identifiers, URLs, classification codes, or regular textual data.

Figure 7 shows a word cloud of all the descriptive tags associated with the CORGIS datasets. This graphic illustrates the range of datasets associated with the project. This also reveals certain biases and trends in the selection of datasets. United States, for instance, is the single largest word in the cloud. Unfortunately, this graphic is not helpful in finding under-represented career paths and disciplines, which is a key problem in the future of the project.

6.2 Surveys

Figure 7: CORGIS datasets word cloud of tags

CORGIS was used in a 50-student Computational Thinking course for non-majors. Students took the course to satisfy a breadth requirement and none had significant prior computing experience. Students completed instructor-assigned practice problems and an open-ended final project where they chose their own dataset from the CORGIS collection.

At the end of the course, a survey was administered to all students as an assignment to gather data on their experience. Four students did not consent to their survey results being shared, and six students failed to complete the survey. This yielded an 80% response rate. Demographically, half of the students were male and half were female. Students came largely from the arts and humanities, with few students from the sciences. The distribution of years was skewed heavily towards Sophomores and Juniors, with only a third of the class being Freshman and Seniors.

Students were asked to rate their agreement with 26 statements on a 7-point likert scale (from "Strongly Disagree" to "Strongly Agree"). The first 25 statements were the cross-product of a pair of five aspects. First were the main course components: learning about abstraction, writing programs, real-world data, social ethics of computing, and working in small groups (cohorts). Next were the elements of the MUSIC model: their belief in whether they had a choice, their interest, their sense of the usefulness, their sense of success, and their belief that the instructors cared. So an example statement would be "I believe that it was useful to my long-term career goals to learn to write computer programs". The last statement, unrelated to the others, was their intent to continue learning computing, either informally (e.g., on online course) or formally (e.g., another Computer Science course).

Figure 8 presents raw results from the survey. Overall, most of their responses were encouraging, showing mostly positive answers (no student marked "Strongly Disagree" for any statement). Students felt empowered and cared for by their instructors. They also indicated that they feel successful in learning the material. We found that their interest in learning course components generally outweighed their sense of the usefulness to learning with respect to their long-term goals. Further, they found working with data related to their own major to be more useful for their career goals than learning to write programs. The lukewarm response to the last statement, students' intent to continue, is expected since it is not the goal of the course to recruit new students into computing. However, we found it disappointing that not a single student marked "Strongly agree" for that element.

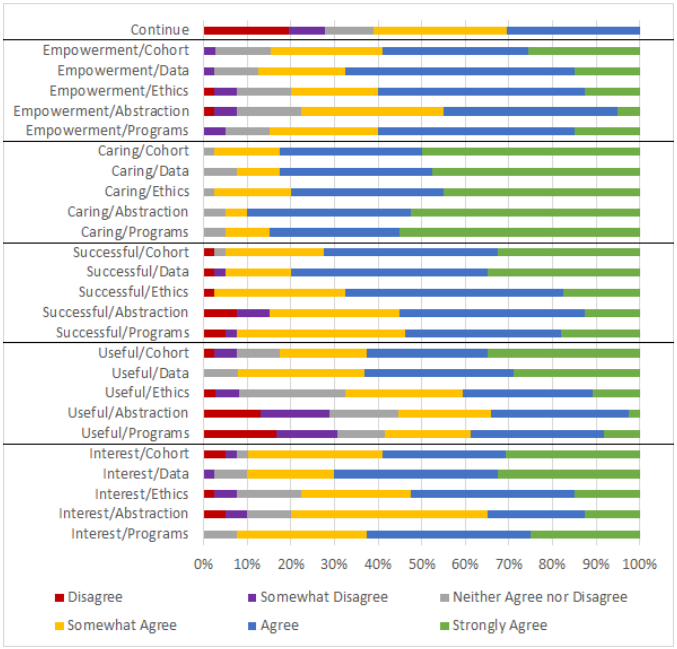

Figure 8: Survey results across motivational and course components

Table 9 reveals a fascinating connection between students' motivation with respect to the content and context and their long-term intent to continue in computing. The vertical categories represent components of the course and the horizontal categories represent elements of the MUSIC model. So, for example, the intersection of the row "Data" and column "I" should be intepreted as "The correlation between students' interest in learning about real-world data related to their major and their intent to continue learning computing". Bolded items are significantly correlated. Although most of the correlations were modest, there is a strong effect between students' sense of the usefulness of learning to write programs and their intent to continue. This correlation does not appear when looking at students' sense of usefulness of learning to work with real-world data. Our interpretation of this result is that, if we wish to encourage students to continue learning computing, we should convince them that learning to write programs can directly benefit their careers. Although the small size of this data does mitigate the finding, and there are a number of other possible interpretations, it does suggest a worthwhile avenue for future research.

	C	M	I	S	U
Abstraction	15.9	26.9	**39.1***	**39.4***	**50.1***
Cohort	14.6	26.6	25.1	28.3	33.2
Data	11.1	16.6	19.8	25.1	30.9
Ethics	17.8	24.4	21.6	**26.5***	**45.3***
Programs	16.3	29.5	**27.8***	**38.5***	**71.1***

Figure 9: Correlation between students' intent to continue vs. components of the course with respect to motivational components (at end of semester)

7. CONCLUSIONS AND FUTURE WORK

The RealTimeWeb and CORGIS projects have been developed over the past 4 years. At this point, it is useful to reflect on the future work needed for the project.

First, further research must be done to evaluate the impact of this type of context on learners' cognition and motivation, similar to the longitudinal studies performed by the Media Computation project [8]. Second, the quantity, quality, and diversity of datasets needs to continue to grow in order to meet the demand of students entering introductory computing. Although understanding the impact on learners is relatively straightforward, expanding the project's collection raises a number of technical issues.

A major limitation of dataset preparation is the expertise required to convert datasets into a format suitable for processing. Although there is significant dedicated research to lowering barriers for this process, it is still the work of a seasoned programmer. Further, every dataset demands some level of domain knowledge to correctly and meaningfully arrange the fields. We also only rely on open-access, unprotected datasets, limiting the field sharply. An exciting but unexplored direction is the creation of artificial datasets based on known characteristics of existing but unsuitable datasets. For example, a detailed dataset representing hospital records would normally be protected by HIPAA. Although such datasets would not allow students to discover trends and facts about reality, we could potentially compensate by embedding a narrative into each dataset. Further research must be conducted to see if this is technically feasible and still perceived as authentic by students.

We hope that interested readers will try out the materials at https://think.cs.vt.edu/corgis, and consider submitting their own datasets to expand our collection.

8. ACKNOWLEDGMENTS

We gratefully acknowledge the support of the National Science Foundation under Grants DGE-0822220 and DUE-1444094.

9. REFERENCES

[1] 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.

[2] 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.

[3] A. C. Bart, E. Tilevich, C. A. Shaffer, and D. Kafura. Position paper: From interest to usefulness with blockpy, a block-based, educational environment. In *Blocks and Beyond Workshop (Blocks and Beyond), 2015 IEEE*, pages 87–89. IEEE, 2015.

[4] D. Burlinson, M. Mehedint, C. Grafer, K. Subramanian, J. Payton, P. Goolkasian, M. Youngblood, and R. Kosara. Bridges: A system to enable creation of engaging data structures assignments with real-world data and visualizations. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 18–23. ACM, 2016.

[5] 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.

[6] 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.

[7] A. E. Egger. Engaging students in earthquakes via real-time data and decisions. *Science*, 336(6089):1654–1655, 2012.

[8] M. Guzdial. Exploring hypotheses about media computation. In *Proceedings of the ninth annual international ACM conference on International computing education research*, pages 19–26. ACM, 2013.

[9] 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*, pages 51–58. ACM, 2006.

[10] 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.

[11] N. A. Hamid. A generic framework for engaging online data sources in introductory programming courses. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2016.

[12] 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.

[13] J. Lave and E. Wenger. *Situated learning: Legitimate peripheral participation*. Cambridge university press, 1991.

[14] 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. 2011.

[15] N. Parlante. Codingbat. *Com (Retrieved 1/08/2011 from http://codingbat. com, 2011)*, 2015.

[16] 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.

[17] L. Torgo. *Data Mining with R, learning with case studies*. Chapman and Hall/CRC, 2010.