Proceedings of The National Conference
On Undergraduate Research (NCUR) 2015
Eastern Washington University, Cheney, WA
April 16-18, 2015

Preliminary Analysis Of Large, Web-Enabled, Dataset For Computing Education

John Doorenbos
Computer Science
Luther College
700 College Drive
Decorah, Iowa 52101 USA

Faculty Advisor: Dr. Brad Miller

Abstract

In studying how students learn and understand computing, large datasets are essential to computing education research, allowing for analysis of educational methodologies on a large scale. Online textbooks are an effective means of gathering large quantities of data which track students' progress through the learning process. This paper discusses the use of a web-enabled dataset to analyze the learning habits of introductory computer science students. It also describes the process of auditing and cleansing the dataset. The short-term goal of the research project was to study how students interact with two online textbooks including *How to Think Like a Computer Scientist* (http://interactivepython.org/runestone/static/thinkcspy/index.html). The online textbooks now experience over 9,000 users a day and have logged more than 29,800,000 events in the two and a half years they have been operational. As a part of the analysis, we compared the student retention rate of these textbooks to that of MOOCs, finding that the online textbooks had similar though improved retention rates. Studying short-term users, we also identified topics that students most often sought help for online. This research project also has the long-term goal of developing a dataset for future research. A major part of this research was auditing the dataset and developing a cleansing workflow that would automatically clean the data. Data auditing included removing data that was anomalous or otherwise unsuitable for evaluation, and standardizing the dataset for ease of analysis. This research is not a stand-alone, completed study; rather, it will assist in making available a large dataset for use by many. The dataset has already been used in a study of the differences in online textbook usage patterns of high-school students, college students, and other online website viewers.

Keywords: Interactive, Education, Data Cleansing

1. Introduction

In order to most effectively teach any discipline, instructors must first understand how their students learn. In the age of information this often translates to having a large dataset which can be analyzed to find trends and patterns in students' learning. Given widespread internet access, web-enabled datasets are an especially effective means of automatically gathering this information.

The Runestone database is a web-enabled database that logs student interactions with the online textbook platform that hosts *How to Think Like a Computer Scientist* and *Problem Solving with Algorithms and Data Structures Using Python* (from here on out referred to as thinkcspy and pythonds respectively). The Runestone database is composed of events that occur anytime the students interact with the textbook. This includes, answering multiple choice questions, highlighting certain parts of the text, running code for an assignment, or stepping through code visualizations. This database, allows for various kinds of analysis. An instructor can follow a single student and view all of his/her activity to try and see that student's flow of thought, or it can be used to see the trends of thousands of students.

This database is not the only one of its kind. At the 2014 SIGSCE conference, an article was published introducing another similar computing education database. The Blackbox Project, launched in June 2013, is a repository of code written by introductory java programmers[6]. Affiliated with the Bluej IDE, the Blackbox Project collects and anonymizes code that can be later used for analysis. Other projects include smaller scale studies, most of which are isolated to individual institutions. The Runestone database differs from the others in that it not only draws data from a large population, and collects source code, but it also records students' interactions with the textbook giving an insight to the entire learning process.

This paper discusses two topics. The first of which covers the process by which the database was filtered in order to make it more suitable for analysis. This includes removal of data that is anomalous or otherwise unsuited for analysis (section 3.1.1), and the normalization of data (section 3.1.2). The second part of the paper looks into the preliminary analyses of the database, covering a basic look at website use and student retention (section 5.2), and a more in depth analysis of short-term users (section 5.1)

2. Background

2.1 Literature Review

As stated before, this is not a new field of study. Research in computing education goes back 30 years to 1985 with a small scale study analyzing buggy pascal code[4]. A 2005 study conducted by Ahmadzadeh, Elliman, and Higgins[5]delved deeply into java compiler errors from students taking introductory computer science courses. Collecting roughly 100,000 error messages over the course of a semester, they analyzed patterns in both compiler and logical errors in novice students. In 2009, Edwards et al.[7] of Virginia Tech compared behaviors of of student programmers and established patterns of behavior common to successful students. All three of these studies have been small scale studies isolated to single institutions. The Blackbox project described in section 1 compares in scale to this study with both have hundreds of thousands of participants from a wide variety of institutions. However, the Blackbox project does not compare in variety of data collected. Collecting data from every part of the learning process, the Runestone database can provide a high resolution look into the learning process of computer science students.

2.2 Textbook Overview

The database used for the project gathers information from textbooks hosted on RunestoneInteractive.org. One of the most widely used online introductory computer science textbooks, it hosts a large range of interactive elements. Because of this, it is particularly well suited to tracking the learning process of students as they use the text. Experiencing over 9,000 users daily, the textbook boasts over 850,000 different readers. The two introductory textbooks analyzed in this project have been used as texts for the first two computer science courses at many colleges and universities including Georgia Tech, University of Kentucky, and Sweden's Royal Institute of Technology. RunestoneInteractive, which hosts textbooks and other educational materials, gives online textbook developers access to a wide range of interactive elements as well as the tools to develop their own interactive elements. The following are examples of interactive elements found in the online textbooks.

2.2.1 codelens

Codelens is an interactive visualization tool that allows students to step through code and displays the global and local stack. Allowing students to step forward and backward through the code, codelens acts somewhat like a debugger. This allows students to start learning what goes on behind the scenes as a python program runs from an stage in an convenient and easy to understand way. Figure 1 shows the codelens interface in action.

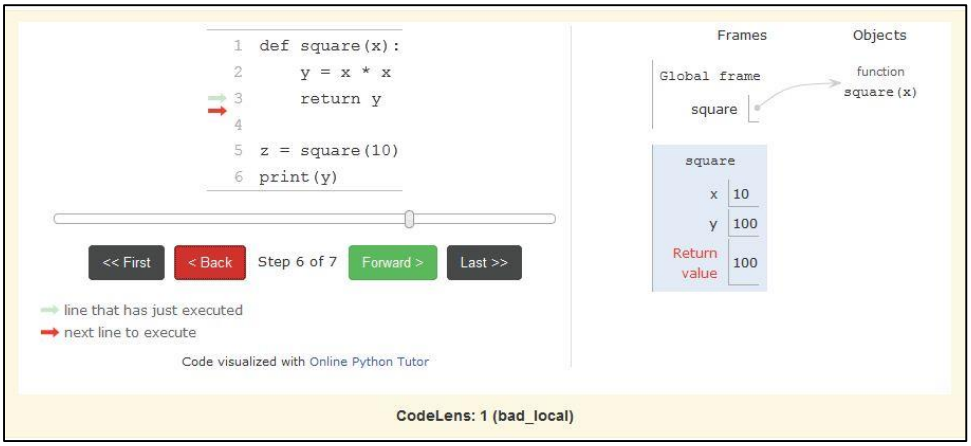

Figure 1: learning about scope through the codelens interface

2.2.2 multiple choice

In order to test students understanding, many sections are concluded with multiple choice questions; multiple choice elements are a concrete way of tracking students' understanding of the material. They also provide feedback if students answer incorrectly, helping clear up student misconceptions about the material.

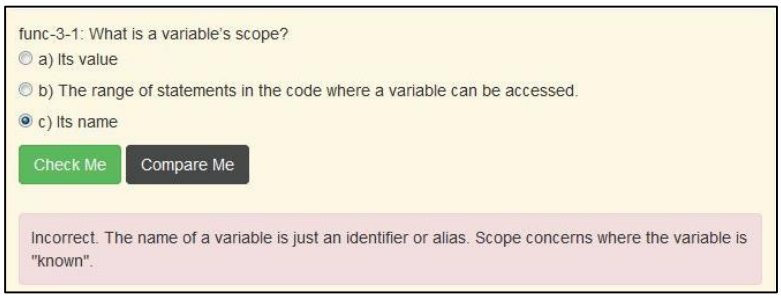

Figure 2: multiple choice element providing feedback to mistaken student

2.2.3 Activecode

Activecode blocks are runnable python examples that give students the opportunity to test and modify code and algorithms described in the text. By embedding runnable code in the textbook, students can complete assignment in a familiar and convenient environment.

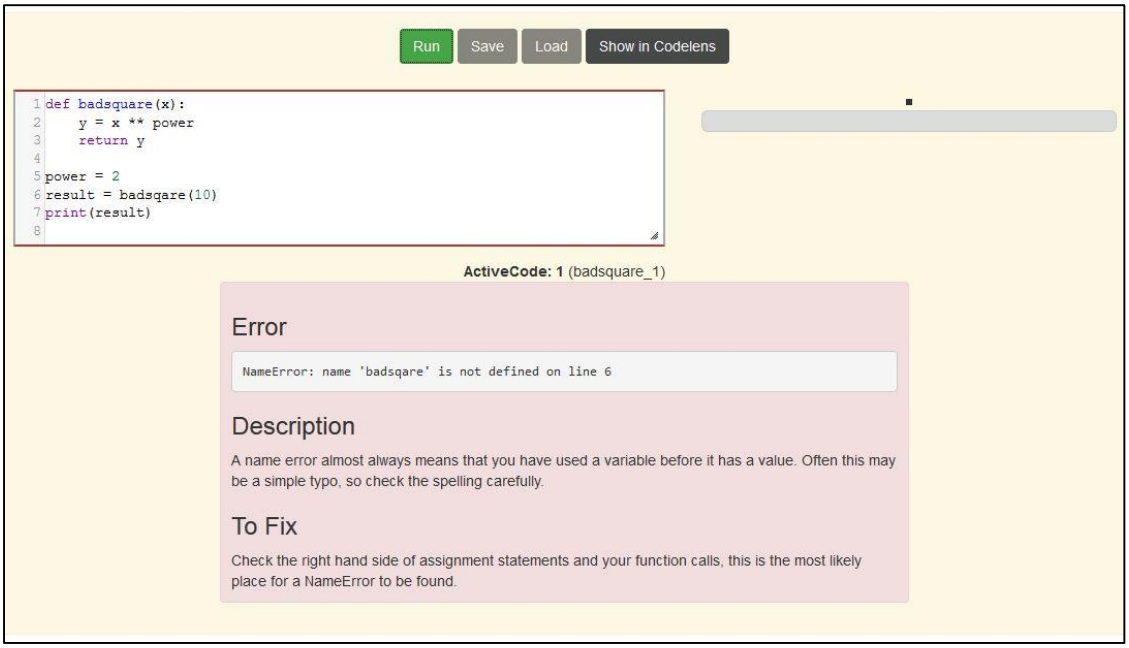

Figure 3: activecode providing detailed error message to student
See Miller and Ranum [1] for a more detailed description of the online textbooks.

2.3 Description Of Dataset

Online since May 23 2012, the database has collected over 29.8 million events and currently experiences over 9,000 users a day. As stated before, it gathers source code that the students write and run as well as the events associated with their interactions with the textbook. Being able to examine the code of novice programmers can give insights to how students are learning specific concepts. Furthermore, having access to the context provided by clickstream information one can start identifying not only what the students aren't understanding, but also why they are making the decisions they are making. While source code written by students was also recorded in the database, this research only investigated clickstream event information.

The dataset in this project comes from the Runestone database. Collecting clickstream event information, it gathers the following information for each interaction:

- **Sid:** a unique student identifier
- **Timestamp:** time of action recorded by server in Central Time Zone
- **Event:** class of item interacted with (i.e. page,codelens, activecode)
- **Act:** specific interaction with item (i.e. page view, activecode run)
- **Div_id:** unique identifier of the element interacted with
- **Course_id:** course that user is associated with

Additionally, the website gathers source code that students write for assignments or in text examples.

3. Methodology

For this project, IPython's Interactive shell and notebooks[3] were used in tandem with Pandas data analysis and visualization module. The dataset described above was accessed using postgresql and Python module sql-alchemy. These tools were used to develop the cleansing workflow described in the sections below. This workflow can be run on any new dump of the dataset to make it suitable for analysis.

3.1 Cleansing Workflow

In order to reduce overhead while logging events into the Runestone database, there is minimal filtering that occurs when information enters the database. This results in a large amount of data that is unsuitable for analysis. In order to generate meaningful analysis, this research focused heavily on cleaning the database. The main goal of this cleanup was to obtain a database that contains only users that exhibit qualities of a student. Data cleanup can be split into two parts: removal of data unsuited for analysis, and normalization of the data. The goal of the data cleanup was to create a database that track the use of students..

3.1.1 data removal

Before data could be removed, minimum requirements for students and courses had to be determined. For a user to be counted as a student they had to be active for more than a day, not registered as an instructor, and interact with more than one unique element in the textbook. Courses were deemed inactive if no user in that course accessed any of the most commonly used chapters for a given textbook. These included chapters on basic data structures, iteration, turtles, and trees. Any data associated with inactive courses was removed. Similarly, some courses were known by the database administrator to be test courses, and were therefore removed.

Even after removing all students and courses that didn't conform to the previously stated requirements, the dataset was still quite noisy. One of the biggest sources of noise was in the div_ids. One category of unforeseen elements cropped up as a result of users downloading pages to their own local machines. As long as their machines were connected to the internet, the database would log their progress. When this occurred, the path to the downloaded chapter was included in the div_id. This can be seen in Figure 4.

	id	timestamp	sid	event	act	div_id	course_id
0	8323469	2013-10-31 08:49:09	1383209349817@172.16.9.14	page	view	/E:/New%20folder/Depth%20First%20Search%20%E2%...	pythonds

Figure 4: an entry in the dataset that contained the path to a user's downloaded page

Another major cause of noise in the dataset was because instructors had the opportunity to modify online textbooks to include their own examples and exercises. This became apparent when trying to calculate user completion in terms of the percentage of elements students interacted with. There were over 3,000 different unique interactive elements in the dataset; far more than expected. In order to avoid as much anomalous noise as possible, a master list of all official div_ids was created. In the master list of elements there were over 600 elements for thinkcspy and over 200 for the pythonds course. The effect of removing courses not in the master list was twofold. It not only cleaned out entries that would be difficult to analyze, but had a normalizing effect on courses in the dataset. The courses could be cross-analyzed as every course contained the same elements. This cross-course analysis is important as it allows comparison of different teaching methodologies.

Finally, in the removal of unsuitable data, outlier users were removed. The first criteria was the users length of engagement in the course. While users with abnormally long active periods were removed, the active periods were measured individually for each book, so users that were active in more than one course could be preserved. Outliers with respect to clicks were also removed. This was necessary as some users dwarfed other users when comparing clicks. This became clear when a few users had orders of magnitudes more clicks that most other users. For example, one university was giving all of its users the same username resulting in a single unique id in the database actually being a conglomeration of hundreds of students. Outlier removal was based on a z-score test and removed users outside 3.5 standard deviations from the mean.

3.1.2 data anonymization and normalization

The final steps in preparing the dataset for analysis included normalizing and anonymizing the data. Over the first two years that the database was operational, some changes to how the data was recorded were made. For the most part this was replacing first year conventions, with second year conventions.

In order to protect the websites' users' privacy, all information was anonymized. This included the sids given to students and the names of courses that students were associated with. Registered users' sids were replaced with "reg_user_xxxxxx", where the tailing x's are a unique 6 digit integer identifier. Independent users, users unaffiliated

with any course, were given the name “anon_user_xxxxxx” following the same naming convention as before. In order to track independent users’ progress, their IP address was saved. Course names were also anonymized in the same way.

4. Results

This section contains preliminary analysis of the dataset, with a focus on gaining a better understanding of student use.

4.1 Short-Term Users

Being a free online resource, the textbooks are available for students unaffiliated with any official course. In fact, much of the interactive textbooks use (~20%) is from short-term users, users active for less than one day. Of these independent users, short-term users were removed from the main dataset into their own dataset for separate analysis. Figure 5 shows the number visits to the website per day from short-term users. Figure 6 shows what pages short-term users were visiting most often (not including the table of contents).

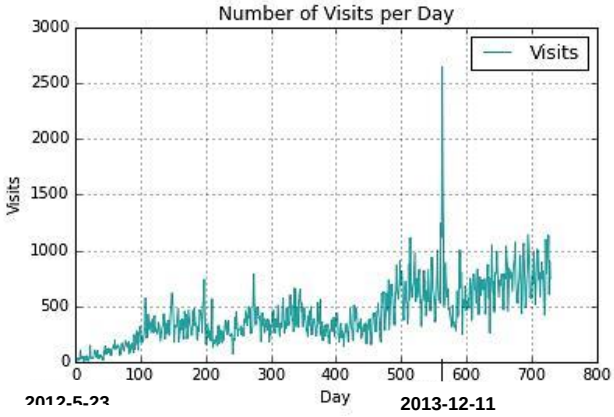

Figure 5: visits per day with spike occurring on December 11, 2013

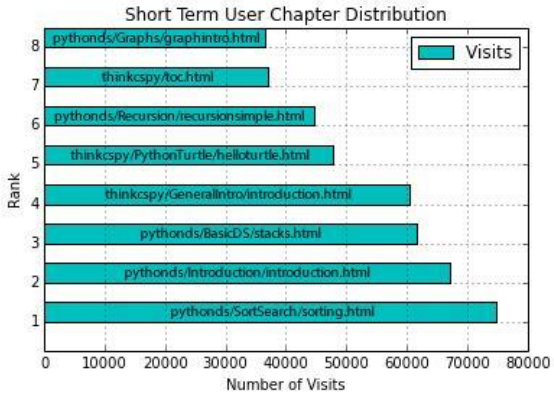

Figure 6: top 8 chapters visited by short term users (does not include table of contents for either book)

4.2 Long Term User Percent Completion

The primary result of interest for this research was investigating the percent completion of long term students, both registered and unregistered, for the two textbooks. Figures 7 and 9 show percent completion rates for both independent users and registered course users, for pythonds and thinkcspy respectively. Percent completion was calculated by dividing the number of unique elements that a student accessed by the total number of unique elements in the textbook being used. In order to see the effects of cleansing the database, Figures 8 and 10 show the percent completion rates calculated with the uncleansed database for both independent users and registered course users, for pythonds and thinkcspy respectively.

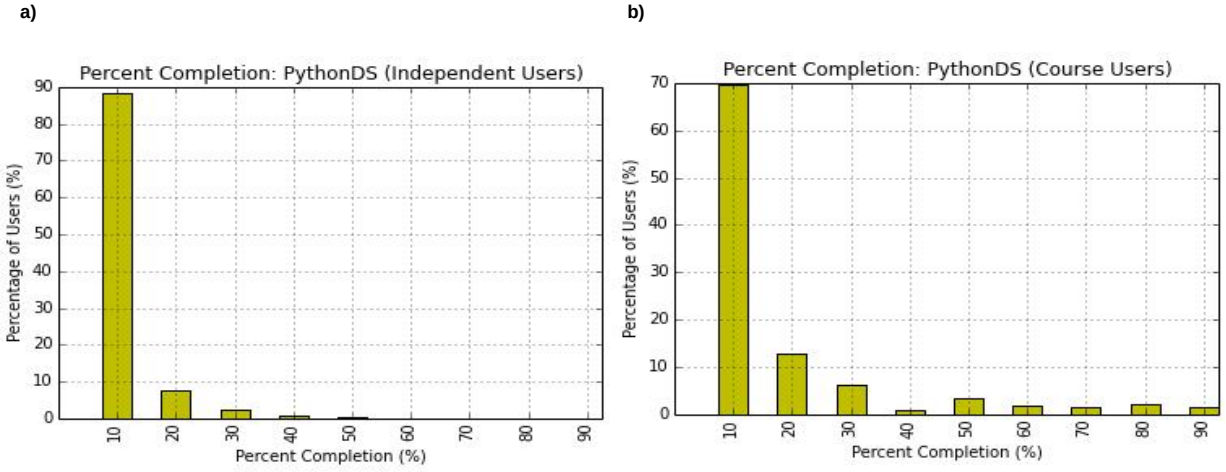

Figure 7: a) percent completion for independent users. b) percent completion for registered users

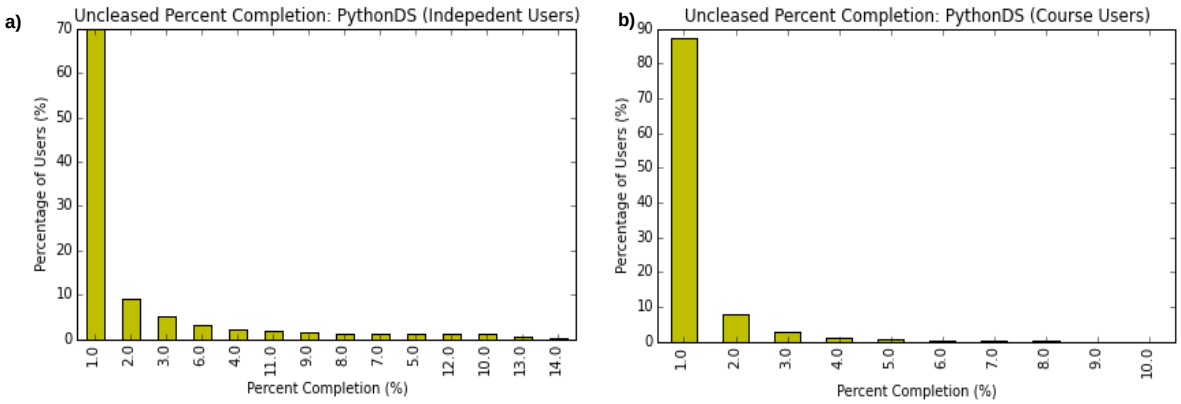

Figure 8: a) uncleansed percent completion for independent users. b) uncleansed percent completion for registered users

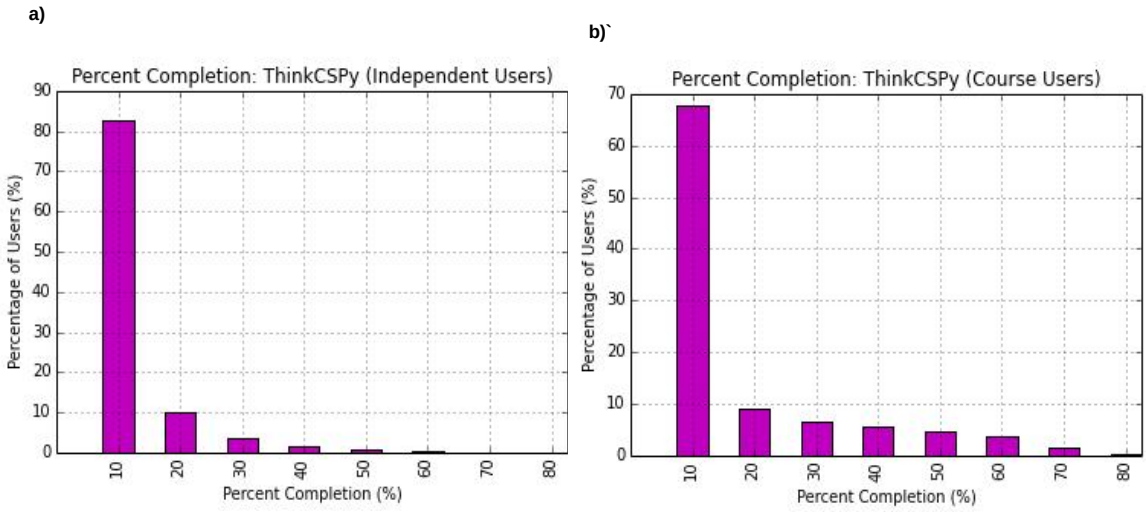

Figure 9: a) uncleansed percent completion for independent users. b) uncleansed percent completion for registered users

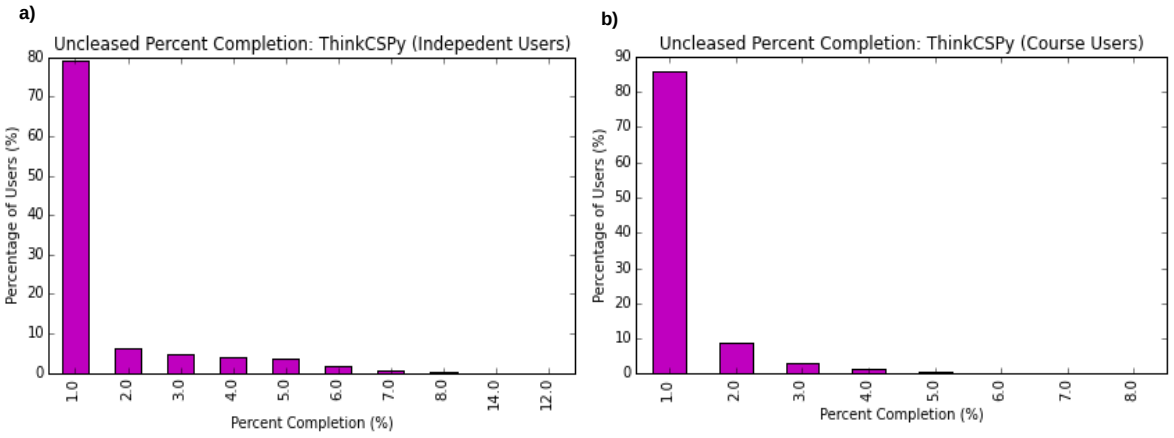

Figure 10: a) uncleansed percent completion for independent users. b) uncleansed percent completion for registered users

5. Discussion

5.1 Short Term User Analysis

While running some basic analytics on the short term users Figure 5 came up as somewhat of an anomaly. When investigating the number of hits on the website as a function of time, a gradual increase in the number of hits per day was expected. Examining the overall trend of the graph, this gradual increase is seen, implying that the website is growing in popularity, allowing the dataset to expand more rapidly as it collects data from more users. However, there is a huge spike in the number of hits on December 11, 2013. With 250% more hits than any other day, there had to either be some error in database or something special about that day. After a fair amount of searching, it ended up being the case that these hits were the result of links to the interactive textbooks being posted on Reddit. This was an interesting case where the dataset reflected the power of social media in the spread of ideas. While the previous finding was interesting, there is much more valuable information that can be gained from short-term users. For example, one can start to infer, based on our dataset, subjects for which students are most often seeking online help. We see in Figure 6 that of all of the chapters (not including the table of contents) sorting has nearly 10 percent more hits any other page. With over 74,000 hits from short term users, users likely only to be visiting for help on a specific

concept, sorting appears to be a subject that students struggle with. With chapters on stacks and turtle graphics also making the top five most visited pages by short-term users, this figure gives insight as to which topics students seek help for online, topics that may be more difficult to understand. This information can be incredible valuable to educators seeking to improve their course. By identifying subjects that give students the most trouble, educators can cater their courses to spend more time on more challenging subjects.

5.2 Long-Term User Percent Completion Analysis

The comparison between Figure 7 and Figure 8, or between Figure 9 and Figure 10 shows the role of data cleansing in providing meaningful analysis. In Figure 8 and Figure 10 we see that, according to the uncleansed dataset, users for both textbooks are completing almost never completing more than 10 percent of the course material. In fact, in the for course users in both pythonds and thinkcspy, nearly 90 percent of students are not completing even 1 percent of the course material. The reason students appear to complete so little is due to the inflated number of div_ids described earlier. Without appropriate data cleansing, it would be nearly impossible to glean any useful information from the dataset. One interest was in comparing the website to other online learning resources, specifically MOOCs (Massive Open Online Courses). With MOOCS commonly exhibiting fairly low rates of students retention, with 91% - 93% of students dropping out before having completed 10% of the course[2], it was of interest to see how these textbooks compared. Figures 7 and 9 show the percentile completion rates of students the two books, pythonds and thinkcspy, respectively. As is seen in Figure 7a, pythonds, the second introductory course, exhibits retention rate similar to that of MOOCs, though slightly higher with just over 12% of students completing 10% of the course. In Figure 7b, we see that users affiliated with the course have higher retention rates, close to 30%, though the course still exhibit a fairly low retention rate. Thinkcspy, the first introductory text seems to tell a slightly different story. As seen in Figure 9a with retention rates at just under 17%completing more than 10% of the book, the textbook seems to more effectively retain students. However, in Figure 9b users affiliated with a thinkcspy course have similar retention rates to pythonds. One possible reason that thinkcspy sees much higher retention than pythonds is that thinkcspy contains more interactive elements. As stated in section 3.1.1, thinkcspy has nearly triple the number of unique interactive elements that pythonds does. While more research must be done on this subject, this could show a positive correlation between the interactivity of a textbook and retention rates. Yet, the question remains: why, even for registered course users, are completion rates as low as they are? One potential answer is that the textbooks are not being used as the primary course materials in classes, but as supplementary materials for students.

6. Conclusion

Like most research, this paper brings forward results; new information to build on a vast expanse of ever-growing knowledge in the field of computing education. However, unlike some research, this project also brings vast opportunity to the table. By providing a large, growing dataset for computing education, this research opens the doors for future research. The raw data, however, has been shown to be unsuitable for analysis, therefore this research has also ensured that the data is properly cleansed to better reflect the students' activity in the courses they are enrolled. There are countless possibilities of potential research that can be done with this dataset. One could group students into different clusters and assess how they learn. Or, one might be able to develop a smart debugger that compares the code of one student to the code of thousands of other student doing the same assignment to give more in depth feedback. This research is not finished; it is the beginning for many new opportunities.

7. References

1. B. Miller, D.Ranum, "Beyond PDF and ePub: toward an interactive textbook." In *Proceedings of ITiCSE '12*. ACM, 2012.

2. D. Yang, T. Sinha, D. Adamson, and R. Penstein. "Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses". In *NIPS Workshop on Data Driven Education.*

3. Fernando Pérez, Brian E. Granger. "IPython: A System for Interactive Scientific Computing". *Computing in Science and Engineering*, vol. 9, no. 3, pp. 21-29, May/June 2007, doi:10.1109/MCSE.2007.53. URL: http://ipython.org

4. J. C. Spohrer, E. Soloway, and E. Pope. “A goal/plan analysis of buggy pascal programs.” *Hum.-Comput. Interact.*, 1(2):163–207, June 1985.

5. M. Ahmadzadeh, D. Elliman, and C. Higgins. “An analysis of patterns of debugging among novice computer science students”. In *ITiCSE ’05*, pages 84–88. ACM, 2005.

6. N. C. e. a. Brown, “Blackbox: A large scale repository of novice programmers’ activity,” in *Proceedings of the 45th ACM Technical Symposium on Computer Science Education, ser. SIGCSE ’14*, , pp. 223–228. ACM, 2014.

7. S. H. Edwards, J. Snyder, M. A. P A. Allevato, D. Kim, and B. Tretola. “Comparing effective and ineffective behaviors of student programmers”. In *ICER ’09*, pages 3–14. ACM, 2009.