Analyzing Pivoting Among Weekly Many Small Programs in a CS1 Course

Joe Michael Allen, University of California, Riverside

Joe Michael Allen is a Ph.D. student in Computer Science at the University of California, Riverside. His current research focuses on finding ways to improve CS education, specifically focusing on introductory programming courses known as CS1. Joe Michael is actively researching the impact of using a many small programs (MSP) teaching approach in CS1 courses. His other interests include educational games for building skills for college-level computer science and mathematics.

Prof. Frank Vahid, University of California, Riverside

Frank Vahid is a Professor of Computer Science and Engineering at the Univ. of California, Riverside. His research interests include embedded systems design, and engineering education. He is a co-founder of zyBooks.com.

Analyzing Pivoting Among Weekly Many Small Programs in a CS1 Course

Abstract

In Fall 2018, our university fully switched from using a weekly one large program (OLP) approach to using a many small programs (MSP) approach in our CS1 course, utilizing a program auto-grader with immediate points feedback and partial credit possible. The switch led to positive results such as an increase in student grade performance and a reduction of student stress. We also saw students making good use of MSPs in their learning, such as spending sufficient time programming each week, and starting earlier on programming assignments. A unique benefit of MSPs is the ability for students to pivot, meaning to switch among programs if they get stuck. This paper investigates such pivoting, and seeks to answer common questions related to pivoting. We analyze how many students pivot and the number of pivots done each week. Given a full-credit threshold (50 of 70 points on 7 programs worth 10 points each with partial credit possible), we examine how students complete the subset of required points. We compare pivot data between a class with a full-credit threshold and a class without. We examine whether students who pivot eventually return to the program from which they pivoted, or if they leave the program unsolved. Finally, we analyze student workflow to observe various pivot patterns. By analyzing student pivot behavior, we hope the community can better understand the pros and cons of pivoting, to help decide whether to adopt an MSP approach and possibly a full-credit threshold.

1. Introduction

Having a positive experience in an introductory programming course, known as CS1, is critical for student success. CS1 is crucial in keeping students in computer science (CS), training non-major students who need some programming, and attracting students to CS. Unfortunately, CS1 courses have many well-known issues: low grades, high stress, low retention, and high drop rates. [1], [2]. A 2014 report by Watson and Li [3] showed that over the past 30 years, CS1 classes have had a rather-high non-passing rate of 30%. Similarly, in 2005, Beaubouef and Mason [4] reported that drop rates between 30%-40% are common for CS programs. These known issues have drawn the attention of education researchers to explore new ways to improve CS1 courses.

One improvement approach makes use of modern program autograders like zyBooks [5], Mimir [6], CodeLab [7], or MATLAB Grader [8]. These modern auto-graders allow for benefits such as giving students immediate, fine-grained feedback, allowing for student resubmissions, and overall saving instructor time on grading and assignment creation. For example, since zyBooks'

auto-grader was released in 2016, over 500 courses (mostly CS1) have started using an auto-grader that did not before. With the benefits these auto-graders offer, many instructors are creating and assigning many small programs (MSPs) per week, rather than the more common one large program (OLP) per week. A previous work [9] summarized a study that showed how MSPs led to happier, less-stressed students, while also improving programming scores on exams, likely due to students having more practice on focused concepts. A later study [10] showed that MSPs lead to students spending good time each week working on their programming assignments, starting to work early on programming assignments, using more programs to study for exams, and a few other benefits. Additionally, that same study also found that using an MSP approach in CS1 did not harm student performance in CS2, in fact, the MSP students performed slightly better in CS2 than OLP students. Since the publication of those studies, our university has fully switched to using MSPs in all CS1 offerings. This paper's purpose is to answer various questions related to the unique benefit that MSPs offer: the ability for students to pivot, meaning to switch from one program to another if stuck.

Section 2 describes our methodology. Section 3 introduces our pivot analysis. Section 4 addresses "How many times do students pivot each week?" Section 5 addresses "What percent of students pivot each week?" Section 6 addresses "What are some observed pivot patterns?" Section 7 addresses "Do students pivot more or less given a full-credit threshold?" Section 8 addresses "Do students return to complete the original program they pivot from?" Section 9 briefly discusses student feedback. Section 10 discusses potential threats to validity. Section 11 concludes.

2. Methodology

2.1 Course

This study was conducted at our U.S. public research university, whose CS department typically ranks in the top 60 by U.S. News and World Report. The university operates on the quarter system. Each academic year is divided into three "regular" 10-week quarters (fall, winter, spring) and one compressed 5-week summer session. Throughout the academic year, the CS1 course serves around 300-500 students each quarter. The course is required for all computing majors and for various engineering, science, and math majors, such that about half the students are computing majors and half are non-computing majors. The course topics include basic input/output, assignments, branches, loops, functions, and vectors. The weekly structure of the course includes three hours of instructor-led lecture, three hours of TA-led labs, interactive online readings, and auto-graded homework assignments. The course teaches C++ as the programming language. The course has a midterm during week six and a final after week 10. Each exam's points come half from multiple choice questions and half from free-response coding questions. The course uses active learning and peer learning in lectures.

2.2 Data collection

We analyzed data from a Winter 2019 CS1 course section that used MSPs. In total, 78 students were in the section used for this analysis. Our CS1 used an online textbook published by zyBooks for all class readings, activities, and programming assignments. At the end of the quarter, we collected all student develops and submits for every programming assignment from the class zyBook. A develop is when a student runs their code through the zyBooks compiler for testing without grading and a submit is when the student "turns in" their assignment for grading. Note that all development was done in the built-in zyBooks coding windows; students were not introduced to an external development environment. Each develop has metadata on the lab title, a chapter section, a userID (anonymized and generated from zyBooks), and a timestamp. A submit has the same metadata as a develop, with additional metadata on the score the student earned on the submission and the max score possible for the lab. In total, we collected data from 78 students for 65 MSPs. We collected 34,316 develops and 14,774 submits for a total of 49,090.

3. Student MSP pivoting

With the MSP approach, students are assigned multiple programming assignments to complete each week. For this class section, students were assigned 7 programs, each being worth 10 points, for a total of 70 possible points in a given week. Students were told that they only needed to earn 50 points of 70 each week to earn full-credit. We refer to this 50 point cutoff as the full-credit threshold. Since students are given a set of programs to complete, they have the unique ability to pivot, or switch among programs while working.

A pivot is when a student partially completes a program (e.g., scores 8 of 10 points) and then chooses to work on a different program. More specifically, a student submit/develop (referred to as a student activity) is defined as a pivot if the activity meets all 5 of the following criteria.

1. The activity is not the student's first activity for the week
2. The activity is for a different program than the previous activity
3. The activity is for a program that has not been completed
4. The previous activity is for a program that has not been completed
5. The activity and previous activity are for programs assigned in the same week

4. How many times do students pivot each week?

Each week, students were assigned 7 programs to complete; each focusing on the topic taught during that week. For example, week 3 teaches while-loops, so students were given 7 programs that focused on loop creation, loop starting/ending conditions, etc. A key question is "How many times do students pivot each week?"

4.1 Analysis and procedure

To answer this question, we first used the rules defined in Section 3 to count how many times each student pivoted during each week of the course. To best understand pivot behavior, for each week, we computed the average number of pivots, the minimum, the maximum, the value of the 1st and 3rd quartiles, and the standard deviation. Finally, we computed an average across all weeks to determine the average times students pivot each week. Only nine weeks were included in our calculations since week 10 has no programming assignments. Students who did not attempt any of the assigned programs for a given week were excluded from weekly calculations.

4.2 Results

Figure 1's box-and-whisker plot summarizes the number of pivots students did each week. Above each whisker are the average number of pivots and the standard deviation. The average number of pivots across all weeks is shown in the top-right corner. The x-axis is the week number and the y-axis is the average number of pivots.

Fig. 1. Box-and-whisker plot to show the pivots each week with a full-credit threshold. The average pivots and standard deviation appear above each whisker (avg, stdev). Total average pivots is 2.2 per week.

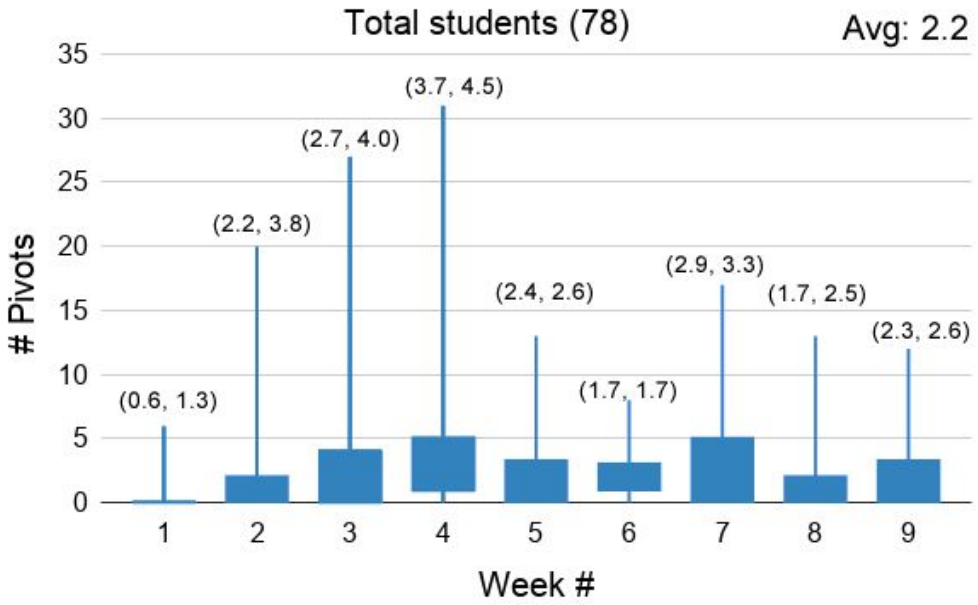


Figure 1 shows students pivoted an average of 2.2 times each week. Week 1 has a much lower pivot rate, due to the programs being easy. Week 4 has the most pivots (3.7 on average), likely due to students being taught while-loops for the first time, one of the most difficult concepts in the course. The standard deviation is larger in weeks 2-4 when students learn expressions, branches, and loops, and lower later when learning functions and vectors.

5. What percent of students pivot each week?

Section 4 analyzed the average pivots each week. Additionally, we wanted to analyze the percentage of students that pivoted each week. A key question is "What percent of students pivot each week?"

5.1 Analysis and procedure

To calculate the average percentage of students that pivoted each week, we first determined how many students pivoted at least once for each week in the quarter. If the student pivoted at least once, they were counted as pivoting for that week. Next, we computed the percentage of students that pivoted each week. Finally, we averaged across all the weeks to calculate the average percentage of students that pivoted during the entire quarter. Students that did not attempt any labs for a given week were not included in the calculations for that week.

5.2 Results

Figure 2 summarizes the average percent of students that pivoted each week. The x-axis is the week number and the y-axis is the percent of students.

Fig. 2. Percent of students that pivot each week.

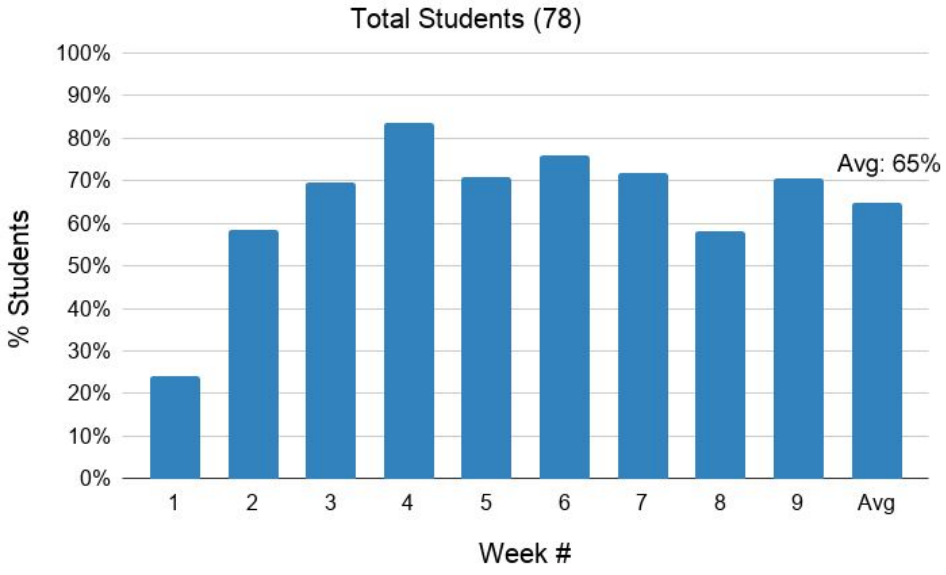


Figure 2 shows that on average, 65% of students pivoted at least once each week. Week 1 has the lowest percent of students that pivoted, likely due to the labs for week 1 being quite easy -- students would complete each without getting stuck. Across the remainder of the term, the percent of students that pivoted seems to be consistent. Across the quarter, 95% of the students in the class (74) pivoted at least once on a program.

6. What are some observed pivot patterns?

Recognizing the ways in which students pivot is important to understand how students utilize the ability to pivot among programs each week. To take a closer look at student pivot patterns, we constructed visual diagrams to represent student workflow. In this section, we show multiple workflow diagrams to visually represent how students worked on their programming assignments during various weeks. A key question is "What are some observed pivot patterns?"

6.1 Analysis and procedure

To visually represent student workflow, we created GANTT charts for each student for every week in the quarter. A GANTT chart shows activities displayed against time. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity [11]. We chose this representation since GANTT charts allow us to see lots of information at a glance. We are able to see the time students spend per program, the total number of develops and submits per program, the score earned per working session, a summary of all activity for the week, and the pivot patterns students displayed. "Ticks" denoted on each line represent a student activity. A tick located above the horizontal bar represents a submit and a tick below the bar represents a develop.

To generate each GANTT chart, we first gathered all student activity for the quarter. From the metadata, using a combination of userID, labID, time spent, and knowing the week each program was assigned, we grouped student activity to do the necessary calculations. To determine time spent, we looked at each timestamp and took the sum of differences to compute the total time spent on each program To compute the number of develops and submits, we counted each activity and if there is a score associated with the activity, then the activity is counted as a submission, otherwise a develop. We calculate the percent scored by finding the highest submission score among all activities for that program. Finally, we separate activity by "workflow blocks," indicating that the student switched working between programs or there was a 10-minute gap observed between activity (10 minutes was chosen arbitrarily). Note, that the reported time spent is thus an understatement as zyBooks does not keep track of activity prior to the first submit or develop, and we exclude time spent calculations if there is an observed 10 minute gap between activity; assuming the student took a break or went to work on something else.

6.2 Results

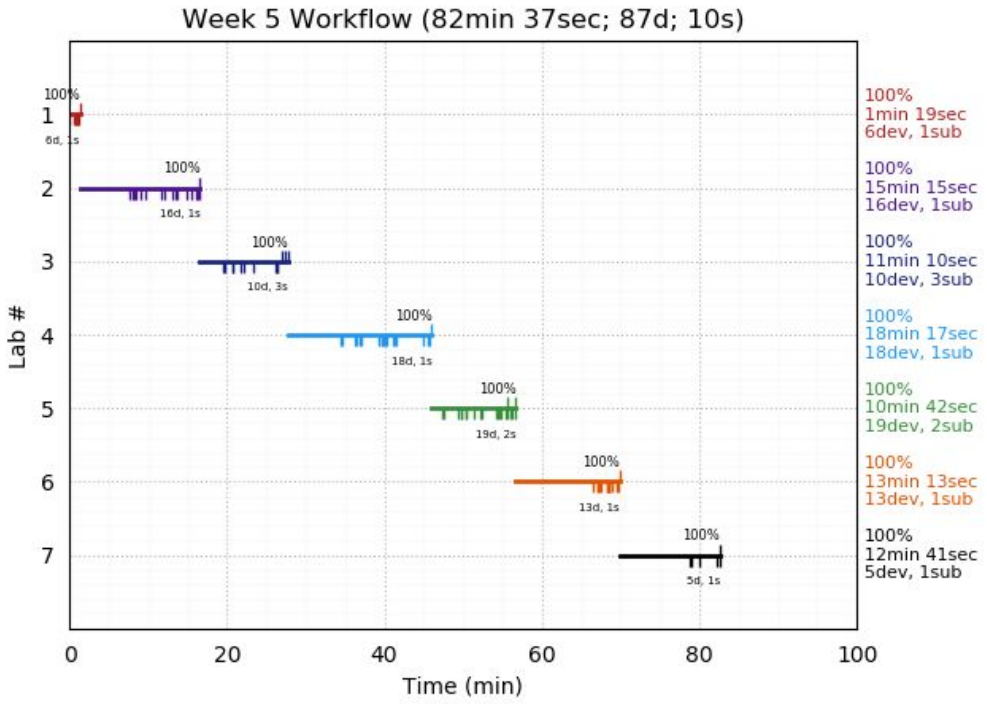
This section presents three different GANTT charts that demonstrate various student workflow patterns. Each graph summarizes the number of submits and develops, the total time spent, and the total score earned. Analyzing a combination of details about each activity also provides

insight on student pivot patterns. Each line in the chart is color-coded to represent a different program for the week. Since students were given 7 programs to complete each week, there are 7 different lines on each chart. The x-axis is the time spent in minutes and the y-axis is the lab number. Note, the time reported in all the charts is an understatement as previously discussed.

6.2.1 Student pattern 1: 0 pivots

Figure 3 shows a student workflow chart from week 5. During week 5, students were being taught for-loops. This student worked straight through all 7 programs, from P1 (program 1) to P7. The student spent the least time on P1, and the most time working on P4. On average, excluding P1, the student spent about 13 minutes working on each program. This student did not pivot while working on this set of programming assignments.

Fig. 3. GANTT chart for a student during week 5.

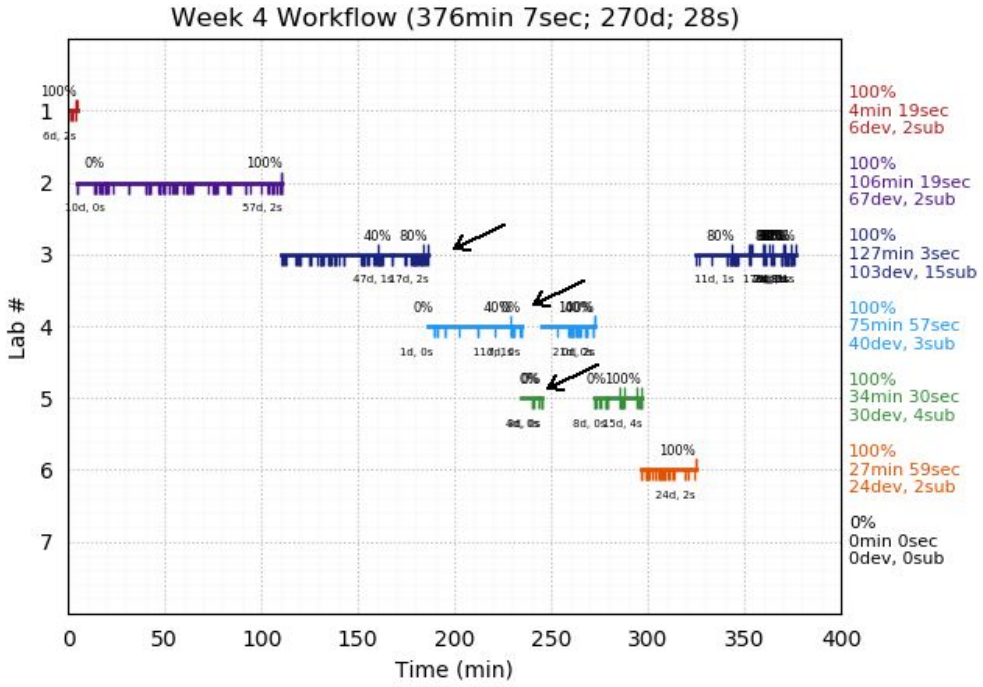


6.2.2 Student pattern 2: 3 pivots

Figure 4 shows a student workflow chart from week 4. During week 4, students were being taught while-loops. This student began working on P1 (program 1) and scored 100%. They moved to P2 and scored 100% after almost 2 hours of working. Next, the student scored 80% on P3 and decided to pivot away to P4. The student struggled with P4, only being able to earn 40% of the points. The student then decided to pivot to P5, but scored 0% the first time around. The student then pivoted and returned to P4 and improved their score from 40% to 100%. The student then returned to P5 and improved their score from 0% to 100% after around 25 minutes

of working. They then completed P6 and returned to work on P3, improving their score from 80% to 100%. The student finally attempted P7, scoring 0 points, and stopped working entirely after that. In total, this student pivoted 3 times; indicated by the arrows in the chart.

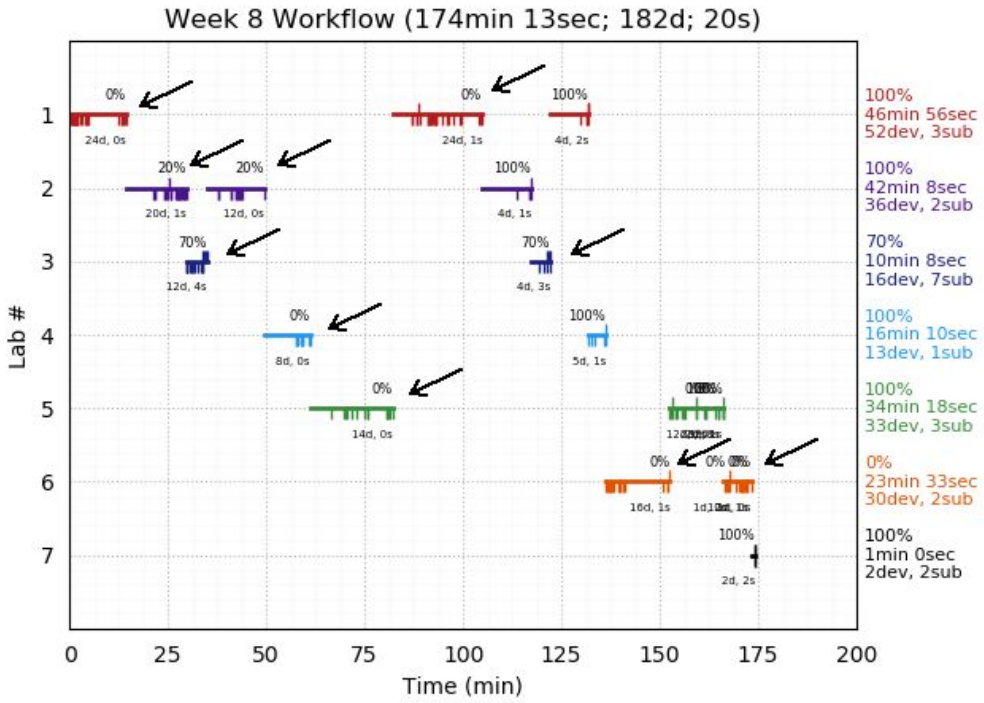
Fig. 4. GANTT chart for a student during week 4.



6.2.3 Student pattern 3: 10 pivots

Figure 5 shows a student workflow chart from week 8. During week 8, students were being taught vectors. This student spent around 3 hours in total working on programs for the week. To summarize, this student spent most of their time working on P1 (program 1), P2, and P5. In total, this student pivoted 10 times during this week's programs. Although the student pivoted and switched many times among the programs, the student was still able to finish and score 100% on most of the programs they attempted. Although the student scored 0% on many of their first attempts, they were still able to pivot between each program and score the needed points to earn 100% for the week.

Fig. 5. Gantt chart for a student during week 8.



These figures provide insight into student behavior when working on programming assignments for the week. These GANTT charts help us understand how students are interacting with the material. Based on these charts, we see that some students do not need to pivot, and can work through all the material straight through, and yet we also see other students make heavy use of pivoting when working on their programs. Overall, we can see that a student may initially struggle, but given time and the ability to pivot, they can learn from other programs and help themselves improve when returning to previously attempted programs for the week.

7. Do students pivot more or less given a full-credit threshold?

The class section used in this analysis was given 7 programs to complete each week (each worth 10 points, 70 points for the week total), but they only needed to complete 70% of the points to get full credit for the week. We refer to the 70% cutoff as the full-credit threshold.

During Winter 2019, two other sections of CS1 were taught without using a full-credit threshold. The other two sections assigned students 7 programs each week, 5 required and 2 optional. Each required program was worth 10 points (50 points total for the week), and the optional programs were worth 0 points (no extra credit). A key question is "Does having a full-credit threshold change pivot behavior?"

7.1 Analysis and procedure

To answer this question, we ran similar analyses as presented in Section 4 and Section 5, but for the other two sections of CS1 offered during Winter 2019. We used these analyses to calculate the average number of pivots each week and the percent of students that pivot when there is not a full-credit threshold. For this analysis, we collected additional data from the other two class sections being taught. In total, we collected an additional 50,655 submits and 91,774 develops from 182 students over 47 MSPs.

Since we collected data from another class section, there are some potential threats to validity. The other class sections did have a different instructor which could lead to instructor bias. Aside from this, all other class variables were kept the same - they used the same online textbook, used a subset of the same MSPs, followed the same course pacing, and took the same exams.

7.2 Results

Figure 8's box-and-whisker plot shows the number of pivots students did each week without a full-credit threshold. Above each whisker are the average number of pivots and the standard deviation. Students without a full-credit threshold pivot an average of 1.6 times each week compared to an average of 2.2 pivots each week by the full-credit threshold students. The x-axis is the week number and the y-axis is the average number of pivots.

Fig. 8. Box-and-whisker plot to show the pivots each week without a full-credit threshold. The average pivots and standard deviation appear above each whisker (avg, stdev). Total average pivots is 1.6 per week.

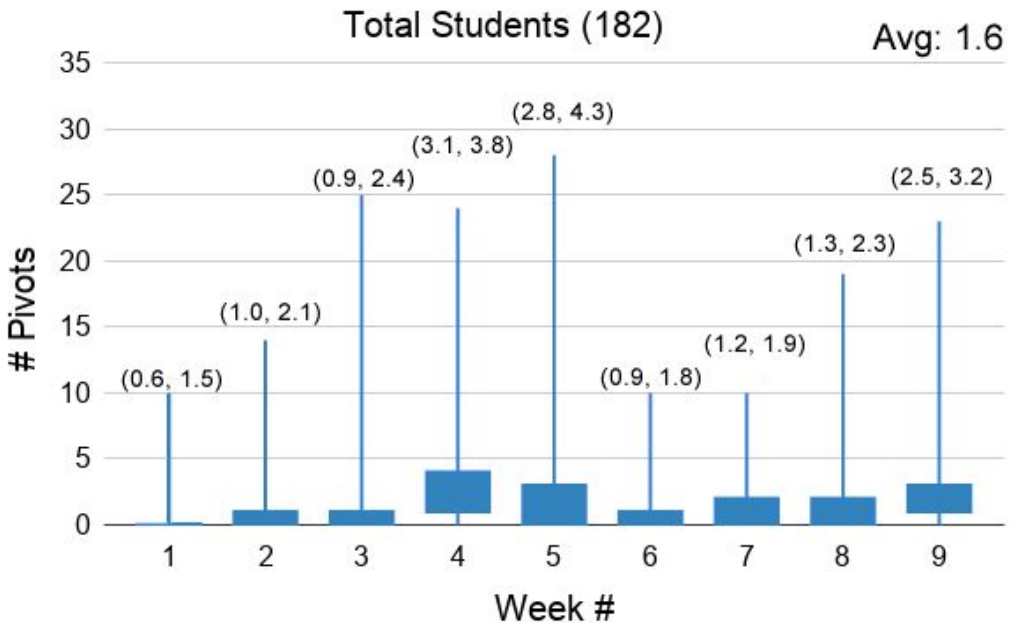
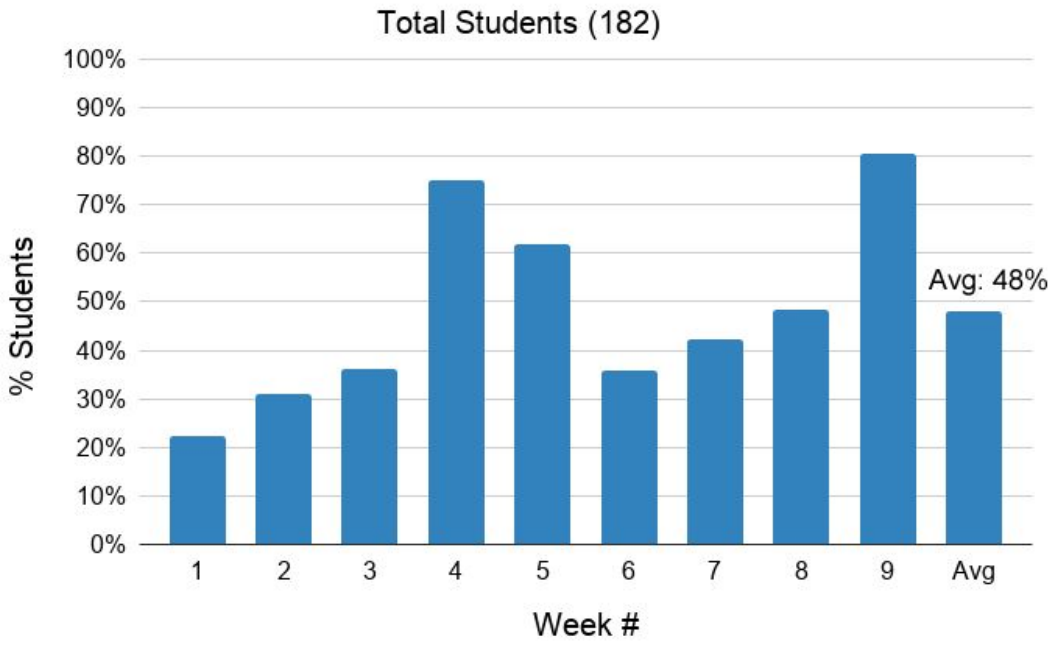


Figure 9 shows the average percent of students that pivoted each week in a class without a full-credit threshold. The x-axis is the week number and the y-axis is the percent of students. On average, 48% of students without a full-credit threshold pivot each week compared with an average of 65% seen by students with a full-credit threshold.

Fig. 9. Average percent of students that pivot each week without full-credit threshold.



Although both groups had the option to complete 7 programs each week, Figure 8 and Figure 9 show that more students pivot, and pivot more frequently when given a full-credit threshold. On first thought, this is likely due to the fact that without a full-credit threshold, students know they need to complete all required programs eventually, so they work through all programs until completion, but more analysis must be done to confirm. One question for further research is "Does giving students a full-credit threshold increase student agency?"

8. Do students return to complete the original problem they pivot from?

One common critique we hear when sharing data on pivoting is that allowing students to pivot could encourage behavior such that students complete the "easy" points of each program, pivot away, and skip the "difficult" parts, thus allowing students to not fully learn programming. To see if this concern is true, one key question is "Do students typically return to the original program they switch from?"

8.1 Analysis and procedure

To address this question, we first define some terminology to categorize student pivot behavior.

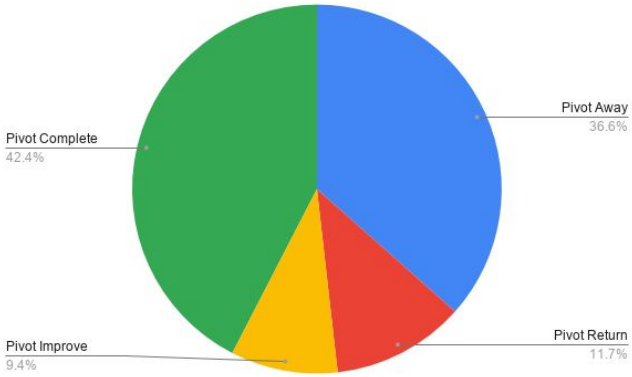
- Pivot none (N): student did not pivot from the program
- Pivot away (P): student pivoted from the program and did not return
- Pivot return (PR): student pivoted, returned to the program, but made no improvement in score
- Pivot improve (PI): student pivoted, returned to the program, and improved their previous score
- Pivot complete (PC): student pivoted, returned to the program, and completed the program fully (scored 100%)

We keep track of all student activity for each program, and then once all activity is completed, we can look at all student activities as a whole to apply the mentioned categories. We isolate the students who pivoted, what they did after pivoting, and if they worked again once they returned. By doing so, we are able to better understand specific pivot behavior.

8.2 Results

In total, students completed 4,596 programs over the quarter. Of those total programs completed, students pivoted on 20% of them. Figure 10 is a pie chart that summarizes pivot categories for the subset of programs that were pivoted from.

Fig. 10. Pie chart summarizing student pivot categories.



To summarize, students returned to 11.7% of programs, returned and improved on 9.4% of programs, and returned to complete 42.4% of programs. Overall, students pivoted from, but never returned to 36.6% of programs. Based on Figure 10, for a majority of pivoted programs, (~65%), students come back to at least submit once more on the program they pivoted from.

Since students return to a majority of attempted programming assignments, this finding shows that students do use pivoting in helpful ways, not harmful to avoid the "hard" parts in a problem. Likely, students use pivoting when stuck to either get ahead on other problems, or they are able to self-enlighten themselves by learning from other programs and then return to work on the programs they were previously stuck on.

9. Student feedback

We surveyed students during week 5 (midway through the quarter) to gather their thoughts on the ability to pivot between programs. Using a 4-point Lickert scale (4 is "Strongly agree", 3 is "Slightly agree, 2 is "Slightly disagree, and 1 is "Strongly disagree") we asked students "I find the ability to jump between programming assignments helpful." The average response was 3.23, indicating that students on average were between "Slightly agree" and "Strongly agree."

10. Threats to validity

10.1 Different instructors

In Section 7, we look at the other sections of CS1 being taught without using a full-credit threshold to compare pivoting behavior between students. Since we collected data from other class sections, there could be an instructor bias as the instructor who taught the full-credit threshold group was different from the instructor who taught the other sections. Although there could be a threat to validity, we note that both instructors are very similar in personality, teaching style, previous evaluations, etc. Both have been teaching for many years together and typically have weekly meetings to share and ensure the class is being run in virtually the same way. Furthermore, all other class variables were kept the same - all sections used the same online textbook, the same programming assignments, followed the same course pacing, and took the same exams.

10.2 Different style of MSPs

In Section 7, we look at the other sections of CS1 being taught during Fall 2019. The other class sections used a slightly different style of MSPs, such that instead of assigning students 7 programs and allowing them to choose which ones to complete for their weekly points, the other class section assigned 7 programs with 5 being required, offering the other 2 for additional practice (but no extra credit). Although these two methods are slightly different, we do not believe this to have much impact on the results of our experiment.

10.3 Outside code development

Although our students were only introduced to the zyBooks in-book IDE, this doesn't mean that students could not use their own IDEs to develop their code outside of zyBooks. If this were the

case, we would be missing some important data on student activity as we would have no way to track their develops and submissions. Knowing this, it is possible that some of our analysis numbers are slightly off. However, since this is an introductory CS1 course, and most students who take this class are new to programming, it's likely that most students did use the zyBooks' in-book IDE primarily to code. Even if this wasn't true, our numbers would be an understatement and we would expect most of our numbers to increase (i.e. more activities could lead to more pivots, time spent, etc.)

11. Conclusion

One way we have tried to improve our CS1 course is the use of MSPs. A unique benefit of MSPs is that students can pivot, meaning to switch among programs when stuck. Since all assigned programming assignments relate to a core topic each week, pivoting is a unique benefit that allows students to gain insight from other programs and then apply that knowledge to solving the previous problems they were stuck on. This paper addressed many common questions about pivoting. We found that students on average pivot 2.2 times each week with a majority of students (65%) making use of pivoting when working on programs each week. We explored various pivoting patterns and saw that students can use pivots to solve problems they previously could not. We showed that students, given a full-credit threshold, do pivot more than students not given a full-credit threshold. Finally, we showed that when a student pivots away from a program, they usually return to work on the program again. There is still much more analysis to be done on the ability to pivot using MSPs, but this work has shown that students are making good use of the benefits that MSPs and pivoting have to offer.

References

[1] P. Kinnunen and L. Malmi, "Why students drop out CS1 course?," in Proceedings of the second international workshop on Computing education research (ICER '06). ACM, New York, NY, USA, 97-108, 2006.

[2] A. Petersen, M. Craig, J. Campbell, and A. Tafliovich, "Revisiting why students drop CS1," in Proceedings of the 16th Koli Calling International Conference on Computing Education Research (Koli Calling '16). ACM, New York, NY, USA, 71-80, 2016.

[3] C. Watson and F. W. B. Li, "Failure rates in introductory programming revisited," in Proceedings of the 2014 conference on Innovation & technology in computer science education (ITiCSE '14). ACM, New York, NY, USA, 39-44, 2014.

[4] T. Beaubouef and J. Mason, "Why the high attrition rate for computer science students: some thoughts and observations," in SIGCSE Bull. 37, 2 (June 2005), 103-106, 2005.

[5] zyBooks. https://www.zybooks.com/catalog/zylabs- programming/. Accessed: March, 2019.

[6] Mimir. https://www.mimirhq.com/. Accessed: March, 2019.

[7] Turing's Craft: CodeLab. https://www.turingscraft.com/. Accessed: March, 2019.

[8] MATLAB Grader. https://grader.mathworks.com/ Accessed: March, 2019.

[9] J. M. Allen, F. Vahid, K. Downey, and A. Edgcomb, "Weekly Programs in a CS1 Class: Experiences with Auto-graded Many-small Programs (MSP)," in Proceedings of ASEE Annual Conference, 2018.

[10] J. M. Allen, F. Vahid, A. Edgcomb, K. Downey, and K. Miller, "An Analysis of Using Many Small Programs in CS1," in ACM SIGCSE Technical Symposium on Computer Science Education, 2019.

[11] Gannt.com https://www.gantt.com/ . Accessed: August, 2019.