Participants: Connor Saari & Jason Kibozi-Yocka
Stakeholder: Jason Zietz, Ph.D.
Workspace: https://github.com/coSaari/ExplorationFinalProject/tree/interation1

Exploration Final Write-Up

The primary problem that revolved around this project was, what worked effectively in INFO 1201? In a broad sense, this question guided us to analyzing and aggregating introductory computing student data in ways that allowed us to make meaningful comparisons as well as helpful tips for future classes. After realizing that we would be working with student data from Dr. Zietz' introductory class, we sought out to find relevant information that would help guide us in our analysis. This outside information came in the form of several very interesting scholarly articles pertaining to this topic. Some of the major points of these findings include failure rates in introductory programming classes, specific topics taught in these classes, and the effects of paired-programming and performance. Interestingly, from the first article, we found that approximately 33% of students pass CS1(introductory programming), and that smaller classes tend to do better than larger classes(have a higher pass rate)[1]. Some specific points from the second topic that stuck out to us was that, according to the results of the study, teaching introductory programming still revolves around the programming language and coding, making students "associate programming with coding" and not with "more abstract, design-oriented and intellectually challenging activities"[2]. Furthermore, in another study of over 20 universities with approximately 1000 students per programming language, the results found that "Python's struggle rate to be significantly higher than that of C++. One possible reason is that C++'s focus on precision translates to a more precise approach to programming"[3]. Finally, our research into paired-programming for intro programming classes offered essential points of interest that could be very relevant to this project. Some of the main points for these studies include finding that students involved in paired programming scored higher on

[1] Jens Bennedsen and Michael E. Caspersen. 2007. Failure rates in introductory programming. SIGCSE Bull. 39, 2 (June 2007), 32–36. DOI: https://doi.org/10.1145/1272848.1272879
[2] Carsten Schulte and Jens Bennedsen. 2006. What do teachers teach in introductory programming? In Proceedings of the second international workshop on Computing education research (ICER '06). Association for Computing Machinery, New York, NY, USA, 17–28. DOI: https://doi.org/10.1145/1151588.1151593
[3] Nabeel Alzahrani, Frank Vahid, Alex Edgcomb, Kevin Nguyen, and Roman Lysecky. 2018. Python Versus C++: An Analysis of Student Struggle on Small Coding Exercises in Introductory Programming Courses. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18). Association for Computing Machinery, New York, NY, USA, 86–91. DOI: https://doi.org/10.1145/3159450.3160586

assignments than their counterparts, and they also had a higher percentage of students who completed the class(92% compared to 76%)[4].

Much of the additional data that we collected revolved around iSnap. This project encapsulates several important data sets that contain programming log data from a visual, block-based programming environment. The primary user base for these data sets were very similar to that of the dataset we received from Dr. Zietz because both included non-major major students for an introductory programming course. Beyond having very similar participant backgrounds and user numbers, these additional datasets also include very similar features. Both have the inclusion of lab grades as well as homework grades which directly correlates to features displayed in Dr. Zietz' data. As for the primary dataset that Dr. Zietz provided, there were many more interesting features to aggregate and analyze. This included features that showed whether students had completed late work for each assignment, ones that showed Python pre- and post-tests, and features that incorporated clicker questioning. Many of these interesting features incorporated in Dr. Zietz' data was very useful for finding some of the preliminary questions we sought to answer, such as, Is there a correlation between late work and one's final grade, how effective are clicker-questions for topic understanding, and do python pre- and post-tests really show a students understanding.

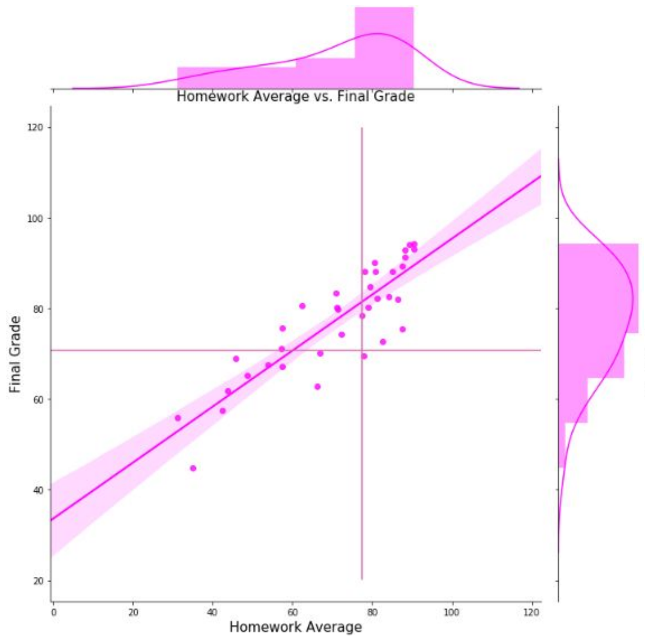
The methods we used to analyze the data we got from Dr. Zeitz were linear regressions, user density models, x and y means, factor plots, and standard deviation. We used linear regression to show us the relationships between final grade and homework average, final grade and clicker grade (aka attendance), and pret-test score versus post-test score. We also used density models on these linear regressions so that we could see where the values were the most concentrated. To supplement these plots, we also plotted x and y mean quadrants to show us which students were above and below the class averages. To help distinguish between students who had done late work and those who hadn't, we used subplots to our benefit by plotting them as two different colors. Additionally, we used factor plots to plot how students did on the homeworks as the semester progressed. We also focused on students who did late work and looked at what their grades would have been if they hadn't done late work.

The overall question we were looking to answer with our analysis was, what would work well in INFO 1201? This question stemmed into several facets of what the features told us about student participation. The first question we sought to answer was how late-work affects student participation and their overall final grade in the class. By aggregating the students into two different datasets, we were able to isolate students who had completed late-work and the students who had completed no late-work

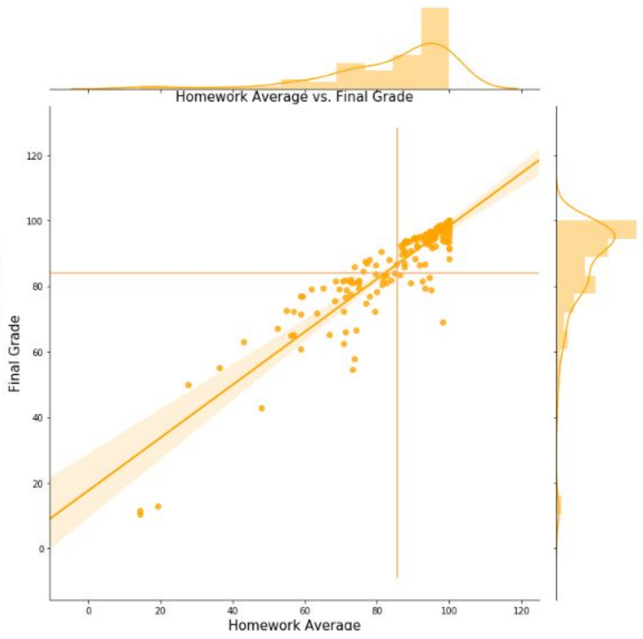
[4] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The effects of pair-programming on performance in an introductory programming course. SIGCSE Bull. 34, 1 (February 2002), 38–42. DOI: https://doi.org/10.1145/563517.563353

through the year. From this, we were able to accurately run some analysis and visualize our findings.

One of the first important pieces of information we decided to analyze was the average grades of the two student groups. Which yielded an average of 77% average final grade for students who had completed late-work, and an average of 85% final grade for students who completed no late-work. This shows that students who took the time to complete and submit late-work ended up with a lower average grade.

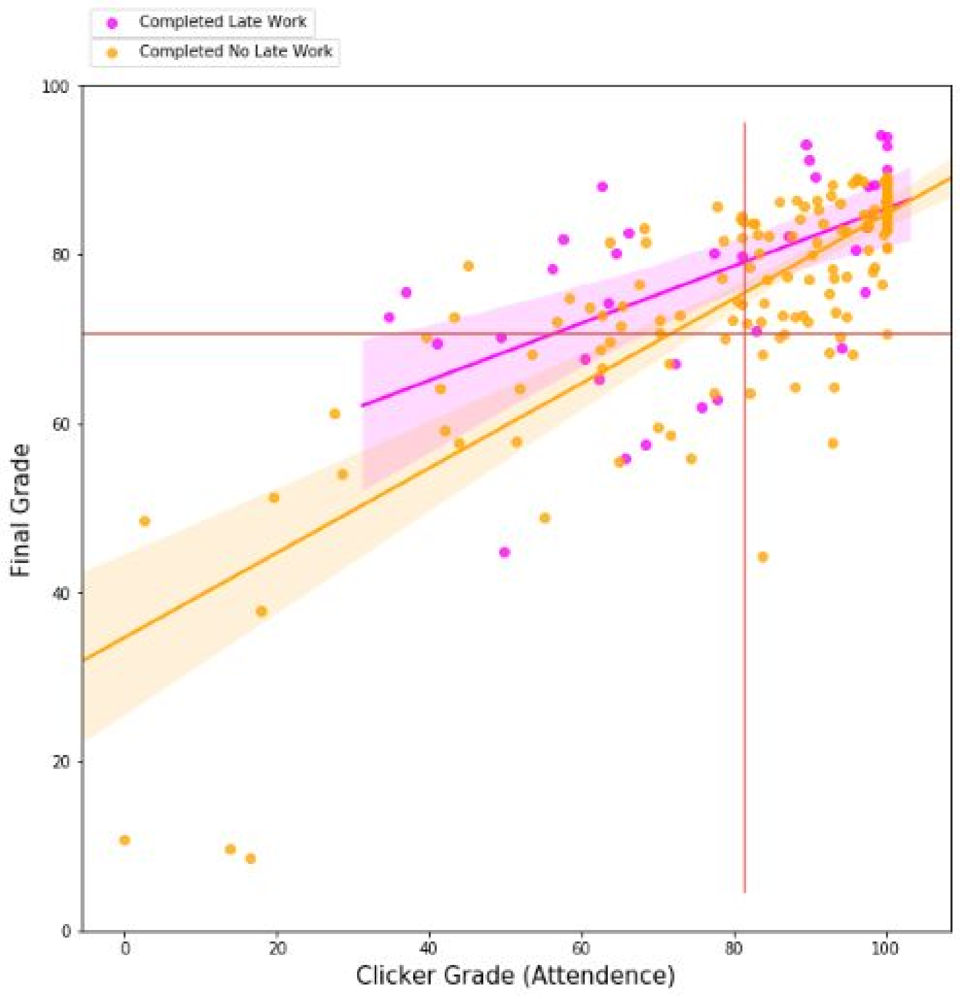


Latework

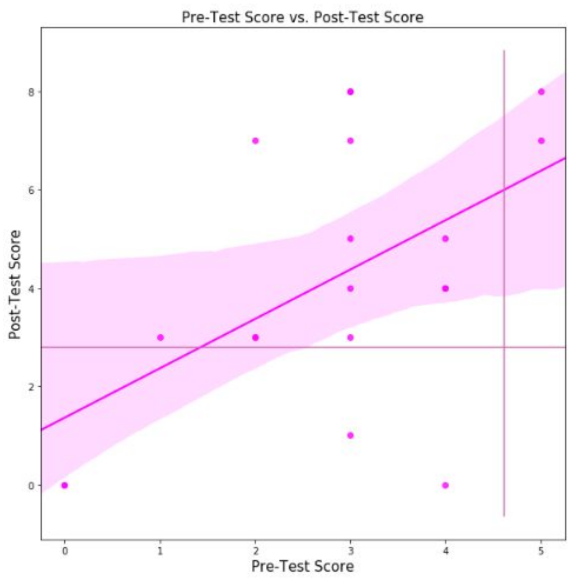


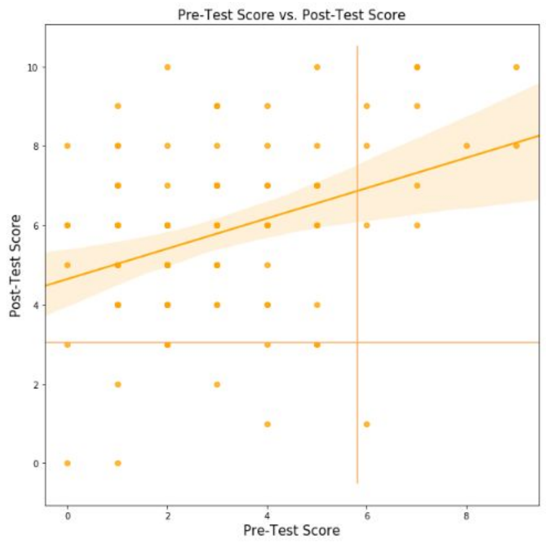
No Late Work

This begs the question, whether or not late-work should be utilized because students seemingly get worse final grades when they are allowed to fall back on late-work. The only real way to find this correlation out is to offer the same class with no late-work, and see if the average grades of the students increase. If students are allowed to fall back on late-work, then their first try might not be completely representative of their abilities because they are relying on that late-work to "bail themselves out". Furthermore, another interesting point that was found during our visualization process was that students who had completed late-work more often attended class. This was based on our third visualization, plotting both groups of late-work and non late-work against attendance (clicker score). This revealed that the average student who completed late work attended class more often than their non late-work counterparts. This can be seen in the two comparative regression lines in the visualization, showing the pink line (late-work) slightly above that of the orange line(no late-work).



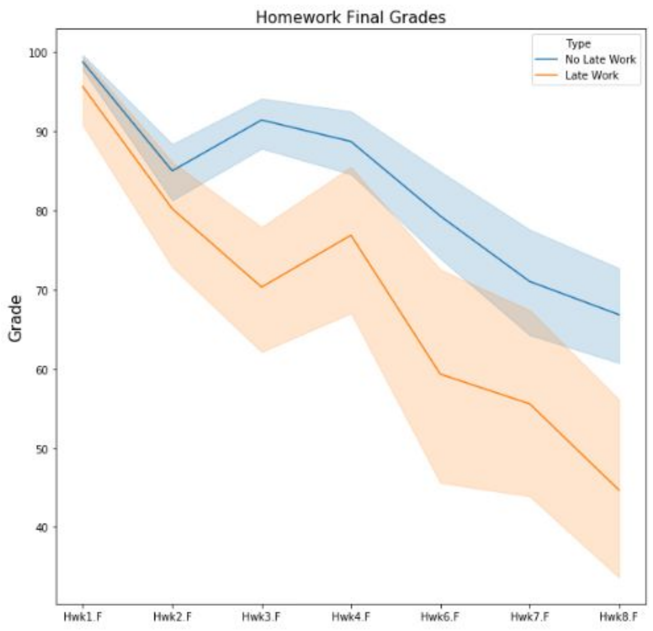
Continuing, another point of interest that was found in the data was that students who had done well on their initial python tests, also did well on their final python test. Comparatively, the students who did poorly on the initial python test also did poorly on the final python test. In the visualizations we displayed the means of the students, showing the different quantiles of students. The top left represents the students who did poorly on the first test, but better on the second, bottom left is students who did poorly on both, and the other side is the same except flipped. Both visualizations of late-work and no late-work showed the

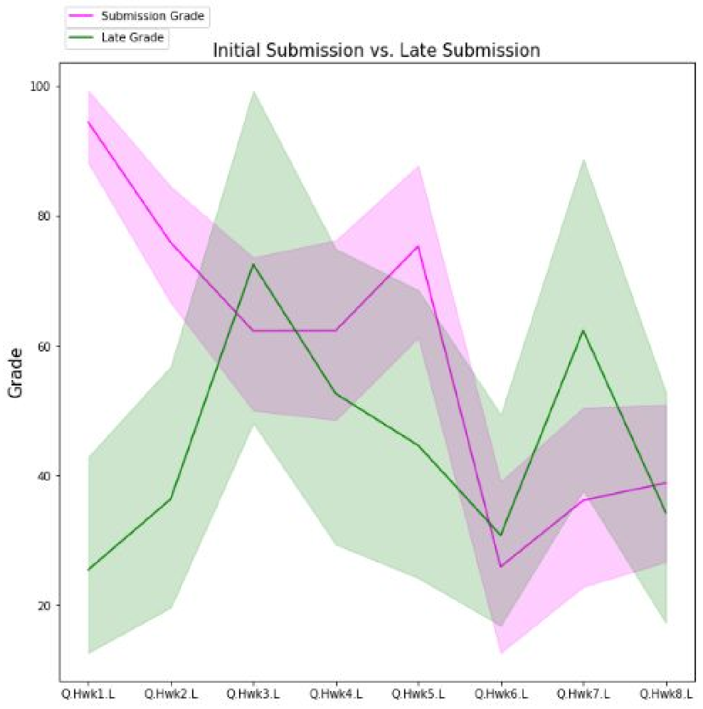




majority of the students were in the top left section, representing good grades on both pre- and post-tests. Overall, this helps us to see that the students coming into this class with coding experience maintained that experience, whereas students coming into this class with poor coding skills left with about the same skill level. This is where we believe the ideals of paired programming might be the most beneficial, because pairing students with relevant computational majors with students who have non-computational majors, may greatly benefit those non-computational major students.

And finally, we witnessed a strange disparity in grades for students who did late work. They, on average, performed worse on homework assignments than students who didn't do late work, despite having done late work, which should have given them a chance to improve their grade. When we dove deeper into the homework grades of the students who did late work, we saw that, on average, they performed worse on their late work than in their initial submission. This tells us that late work only seemed helpful for the harder assignments like Homeworks 6 - 8, which the whole class did poorly on.





Did Late Work

All of our analysis will be linked with this final write up in the form of a github link. This final github will encapsulate all of the relevant files we used and the corresponding jupyter notebooks we utilized for our analysis. Beyond this, we have a couple of recommendations for INFO 1201. As for late-work, there are some interesting things to think about. At the bottom of the primary jupyter notebook, the average grades for both groups fell toward the end of the semester, this correlated to a spike in late-work. This is where late-work was the most utilized and effective, primarily because we think grades dipped towards the end meaning more reliance on late-work submissions. We think the most effective way to utilize late-work in this class would be to not tell students you offer late work until the end of the semester. This would put more emphasis on completing homework submissions the first time around and not being able to rely on the late-work to fall back on. By only offering late work at the end of the semester, students will be given leniency during finals period relieving stress and helping their overall grade, as well as revisiting topics from earlier assignments possibly affecting the outcome of the python post-test. Another very interesting point to be made is the effectiveness of paired-programming. When looking at the comparison between the python pre- and post-tests, we believe it would be the most beneficial to utilize paired-programming for greater topic understanding. This would be beneficial for those students who do not come from computational backgrounds and are majoring in other non-related subjects. By pairing these students with those who have a computational background, the collective knowledge of topics understood in the class will yield much better results. Generally, by offering paired-programming and late-work submissions at the end of the year, the class as a whole will benefit.