A Case of a Computational Setting Facilitating Empirical Re-Conceptualization

Adaline De Chenne
Oregon State University

Elise Lockwood
Oregon State University

The use of computer programming is ubiquitous for research mathematicians, and two common practices are facilitation of experimentation and testing of conjectures. These practices align with empirical re-conceptualization, which is the process where empirical data is used to identify patterns, form related conjectures, and then re-interpret the conjectures from a structural perspective. However, literature on empirical re-conceptualization has only examined student work done by hand. We present case study data of one student, Allen, using the programming language Python to facilitate empirical re-conceptualization as he found the closed-form solution for binomial coefficient C(n,k). *We discuss why the computational environment was productive for empirical re-conceptualization and provide avenues for future research.*

Keywords: Computation, Re-Conceptualization, Combinatorics, Generalization

Introduction, Motivation, and Relevant Literature

Mathematics as a discipline has adopted computation to progress research that has historically been performed by-hand, to provide additional insights into existing theory, and to pursue new solution methods for difficult problems. As computation—which, for our purposes, refers to the use of a computer to carry out, or otherwise assist in, a solution to a mathematical task—becomes increasingly ubiquitous in the discipline, there has been a push by mathematics education researchers for investigations into the interplay between computational environments, mathematics, and students. This has included investigating how computational environments can be used to facilitate conceptual development in novel and complementary ways (e.g., Sand et al., 2022), how to assess student use of computation (e.g., Buteau & Muller, 2017), how computer programming informs student affect towards mathematics (e.g., Purdy & Lockwood, 2020), and how further research might benefit from theoretical perspectives that accommodate the increasing relevancy and diffusion of computing into mathematics (e.g., diSessa, 2018). Computing is now an authentic mathematical practice (Lockwood et al., 2019), and as mathematics education research seeks ways to foster student development of authentic mathematical practices across subject areas, we feel there is an opportunity to do so for computing. Specifically, Lockwood et al. (2019) identified ways in which mathematicians use computing in their work. In this paper, we build on two possible ways students can use a computational environment that align with those identified in Lockwood et al., (2019): to facilitate experimentation, and to test conjectures.

Experimenting and conjecturing align with the concept of empirical re-conceptualization (ER) (Ellis et al., 2022), which is "the process of identifying a pattern or regularity, forming an associated generalization, and then re-interpreting those findings from a structural perspective" (p. 2). As yet, the literature on ER has only examined student work performed by hand. This work includes either the researchers providing a data set for students to study and conjecture about, or the students creating the data set themselves. We feel that a computational environment can facilitate ER by providing the means to quickly produce large data sets in ways that are intractable by hand, and to test the veracity of conjectures over these data sets. Moreover, this provides an opportunity to investigate how a computational environment can be used in

problems students are capable of solving while still having them engage in practices that are authentic to the discipline.

In this paper, we seek to demonstrate an instance of a computational environment facilitating ER for a student, Allen (a pseudonym), as he finds a closed-form expression for the binomial coefficient, *C(n,k),* by writing computer code in Python. We will show that Allen used the computational environment to create data sets based on the parameters *n* and *k*, and he used the data sets to iterate between creation and validation of conjectures. In doing so, we hope to contribute to the literature on computation in mathematics education, and to the literature on conjecturing and generalization, by documenting one way that a computational setting can be used to open the door for students to engage in mathematical generalization. We also contribute to the literature on combinatorics education by discussing how Allen's counting process for *C(n,k)* involved enumerating and tallying the entire set of outcomes, and discussing how he had difficulty connecting his counting process to the closed-form expression. This paper addresses the research question "In what ways might a computational setting facilitate empirical re-conceptualization (ER) as students find closed-form expressions for binomial coefficients?"

Theoretical Perspective on Empirical Re-Conceptualization

Curricular development and mathematical reasoning hierarchies expect students to move from empirical toward deductive reasoning, and they broadly frame empirical reasoning as less sophisticated than deductive reasoning, or as a stumbling block to overcome. This is justifiable, as empirically-found patterns can be faulty, lack explanatory power (Bills & Rowland, 1999), and students may overly rely on a small number of cases to check the validity of mathematical statements (Knuth et al., 2009). Yet, students can be adept at empirical patterning (Küchemann, 2010), and they demonstrate dynamic interplay between patterning empirically and arguing deductively, just as mathematicians do (de Villiers, 2010). Because students have difficulty shifting from empirical to deductive reasoning (e.g., Stylianides & Stylianides, 2009), there is motivation to search for ways of leveraging the productive aspects of empirical reasoning in order to enhance, or otherwise inform, deductive reasoning. As an example, empirical reasoning can give insight into a problem's structure in a way that supports deductive reasoning. Further, students flexibly reason between multiple representations when reasoning empirically (Amit & Neria, 2008), which can be beneficial for deductive reasoning. Ellis et al. (2022) propose the concept of ER as one way that empirical reasoning can be leveraged to enhance deductive reasoning.

Empirical re-conceptualization is "the process of identifying a pattern or regularity, forming an associated generalization, and then re-interpreting those findings from a structural perspective" (Ellis et al., 2022, p. 2). A structural perspective refers to the last four categories of structural reasoning as described by Harel and Soto (2017), whose five major categories are (a) pattern generalization, (b) reduction of an unfamiliar structure into a familiar one, (c) recognizing and operating with structure in thought, (d) epistemological justification, and (e) reasoning in terms of general structures. The first category further elaborates the difference between result pattern generalization (RPG), which is a way of generalizing that attends only to results, and process pattern generalization (PPG), which is a way of generalizing that attends to a process (Harel, 2001). Generalizations formed through empirical reasoning are examples of RPG because only the results (i.e., the empirical data) are used to create the generalization. A structural perspective refers to the other four categories of structural reasoning. ER can thus be evidenced by a shift from RPG to PPG, which entails a shift from pattern generalization to any of

the other four categories. In this study, we use this notion of ER to frame our student Allen's work, and we highlight the ways in which the computer supported Allen's ER.

Methods

Participants and Data Collection

The data presented here are from a teaching experiment (TE) (Steffe & Thompson, 2000) that consisted of 3-4.5 hours of contact time each with four students in 60- to 90-minute interview sessions over the course of several weeks during a single academic term. This TE is part of a broader study that aims to investigate how a computational environment can be leveraged to enrich student understanding of mathematics. We focus on a single student, Allen (pseudonym), who was an Electrical and Computer Engineering major recruited from an introductory computer science course at a large university in the western United States. We chose Allen based on his responses to a recruitment questionnaire, which indicated that he had never taken a discrete mathematics course, and he exhibited a strong computer science background. We interviewed Allen in person over three 90-minute sessions, and we focus on the last session in this paper. We chose these data because they exemplify using computational techniques to engage in ER through experimentation and conjecture validation in a sophisticated way. Indeed, Allen's mastery of fundamental computer science ideas indicates that most students would not produce similar results, but we nevertheless believe our findings have useful theoretical implications about using computer programming to engage in ER. For example, our findings might speak to how future studies could create tasks and scaffolding intended to elicit generalizing activity, such as ER, in a computational setting.

During the TEs, the students sat at a desk and worked individually on a desktop computer in the programming environment CoCalc. CoCalc is an online-supported application that allows users to write in Jupyter notebooks, which include cells where Python code can be written and executed. Jupyter notebooks allowed the students to write separate pieces of code for each task, or multiple pieces of code for a single task, and to quickly reference code they had previously written. We also gave them paper handouts of the problem statements and provided a brief introductory guide to Python syntax. To capture the interviews, we videotaped and audiotaped the interviews, took screen recordings of the desktop computer, and scanned the students' hand-written work.

Tasks

Allen worked on a number of counting problems that involved finding the solution by hand, and writing a computer program that enumerates and tallied all possible outcomes. Throughout the TE, he worked on problems involving Cartesian products, arrangement with repetition, arrangement without repetition, and one problem involving selection without repetition. Allen used the multiplication principle to solve these problems by hand (with the exception of the selection without repetition problem), and his computer programs used nested *for* loops to implement an odometer strategy, similar to those reported by De Chenne and Lockwood (2022).

We focus on his work on the Book Problem: "Suppose you have eight books and you want to take three of them with you on vacation. How many ways are there to do this?" The solution to this problem is $C(8,3) = 56$, the number of ways to *select* (without arrangement), three of eight books. We chose these data because Allen initially solved the problem incorrectly, and after finding the correct solution he decided to search for patterns between his initial incorrect answer and the correct answer. The pattern he found made him curious about how the pattern would

change with different parameter values, and Allen engaged in ER ultimately to find a closed-form expression for the binomial coefficient, *C(n,k)*. We elaborate on Allen's solution, and his ER, in the Results section.

Data Analysis

We chose this episode because it demonstrates an instance of a computational setting facilitating ER. The first author identified key moments where Allen used his computer programs to reason empirically about the relationship between *C(n,k)* and *P(n,k)*. She then created an enhanced transcript of the episode by including pictures of Allen's work and screenshots of the computer programs he wrote. This analytic process allowed us to make a narrative of Allen's reasoning about *C(n,k)* as it relates to *P(n,k)*, and capture instances of conjecturing, validating conjectures, and re-conceptualizing the empirically-found patterns. Finally, both authors reviewed the enhanced transcript and identified ways in which Allen's work with the computer demonstrated instances of ER.

Results

We first provide a chronological narrative of Allen's work finding a closed-form expression for the binomial coefficient, *C(n,k)*. Then, we discuss three ways in which the computational environment facilitated the ER: (1) isolating generalizable relationships, (2) generating data, and (3) validating conjectures.

Allen began his work on the Book Problem (stated above). The correct solution to this problem is *C(8,3)*, the number of ways to *select* three of eight books, but Allen's initial solution was *P(8,3) = 8*7*6 = 336*, the number of ways to *arrange* three of eight books. The first author asked him how he would verify if his solution was correct, and Allen verbally described a computer program that would enumerate the outcomes. The computer program he described enumerated all *arrangements* of the books, which was consistent with the numerical solution he found, but was incorrect because it overcounted the outcomes. The first author asked him to write down the first ten outcomes his computer program would produce. When he wrote down these outcomes, Allen noticed that the computer program he described would produce both *1,2,3* and *1,3,2*, which represent the same outcome. Based on his realization that *1,2,3* and *1,3,2* represent the same outcome, we infer that Allen understood that different arrangements of the same three books do not constitute different outcomes in this problem.

After realizing his error, Allen was unable to find a closed-form expression that correctly counted the outcomes, but he was able to write a computer program that would correctly enumerate all the outcomes. To do so, he modified his enumeration strategy for arrangements so that a length three number sequence was enumerated only if the numbers in the sequence were in increasing order, as seen in Figure 1. This modification ensures a unique order for any length three number sequence, and so each combination is only counted once. Allen justified this modification by stating "Say I pick book number 5 as my second book. I don't need to pick book 1, 2, 3, 4, or 5 [as the third book] in that case, so I'd only have three books to choose from." After executing his code, Allen found that the correct answer was one sixth of his original, incorrect answer; that is, Allen found that *P(8,3) = 6*C(8,3)*.

After finding that his initial solution was six times greater than the correct solution, Allen stated

> *Allen*: That's interesting, *8*7* is *56* … Completely the third value [the value *6* in *8*7*6*] didn't even really matter it looks like … I just want to see if bringing the total number of books down by one, does that make it equal *7*6*? Is there a relationship there?

This is the first instance where Allen reasoned empirically about a generalizable pattern. He noticed that his initial solution was *8*7*6* and the correct solution was *8*7*, and he conjectured that *C(7,3)=7*6*. We infer that Allen's conjectured pattern was that the correct answer could be found by first finding his original answer, and then removing the last term in the product. Hence, because *P(7,3) = 7*6*5,* Allen conjectured that *C(7,3) = 7*6* because the term *5* would be removed. This is an instance of RPG because Allen found the numerical pattern based solely on the results, and he did not provide structural reasoning that justified the pattern, or connect the pattern to his computer program. Allen modified his computer program to find *C(7,3)*, seen in Figure 1, and he found that *C(7,3) = 7*5*. After observing that *P(7,3) = 6*C(7,3)*, and the ratio was not *5* as he anticipated, Allen conjectured that *P(n,3) = 6*C(n,3)* for every value of *n*. This conjecture is another example of RPG because he justified the pattern based only on the observed results. To validate this conjecture, Allen wrote a function that would return the ratio for an input value of *n* (which is called 'book' in his code), and he used the function to check that the ratio was *6* for all *n* values up to 21. After validating this conjecture for those *n* values, Allen concluded that the ratio was *6* for all *n* values.

```
count = 0

for i in range(1,9):
    for j in range(i+1,9):
        for k in range(j+1,9):
            count = count + 1
            print(str(i)+" "+str(j)+" "+str(k))
print count
```

```
count = 0

for i in range(1,8):
    for j in range(i+1,8):
        for k in range(j+1,8):
            count = count + 1
            print(str(i)+" "+str(j)+" "+str(k))
print count
```

```
count = 0

for i in range(1,7):
    for j in range(i+1,7):
        for k in range(j+1,7):
            count = count + 1
            print(str(i)+" "+str(j)+" "+str(k))
print count
```

```
def fun(book):
    count = 0
    for i in range(1,book+1):
        for j in range(i+1,book+1):
            for k in range(j+1,book+1):
                count = count + 1
                #print(str(i)+" "+str(j)+" "+str(k))
    return (book*(book-1)*(book-2))/count
```

Figure 1: Allen's computational solutions for C(8,3), C(7,3), C(6,3), *and* C(n,3).

Allen then conjectured how the ratio of *P(n,k)* to *C(n,k)* would change for other values of *k*. To generate a conjecture, Allen observed that *6=2*3*, and he connected the *3* in the product to the parameter value *3* in *C(n,3)*. He conjectured that *P(n,4) = 12*C(n,4)* because *12 = 4*3*. This is another instance of empirical patterning, as Allen did not provide, or search for, structural justification for why the parameter value *3* was connected to the ratio. To validate his conjecture, Allen modified the function he previously wrote for *C(n,3)* to calculate *C(n,4)*. This modification did not appear to be difficult for Allen, and he made the appropriate changes without prompting by the researchers. Using this function, Allen found that *P(n,4) = 24*C(n,4)*, which was not the ratio he anticipated. He then characterized the change in ratio from *6* to *24* as a multiplication by 2^2, not as multiplication by 4, which we take as further evidence that Allen was empirically patterning based on the *2* in the original product *6 = 2*3*. Allen then conjectured that *P(n,5) = 40*C(n,5)*, where the $40 = 2^3*5$. This is another example of RPG, where Allen generalized the results (factors of *2*) without structural justification. However, after further modification of his code to calculate *C(n,5)*, Allen found that the correct ratio was *120*.

At this point, Allen had found that *P(n,3) = 6*C(n,3)*, *P(n,4) = 24*C(n,4)*, and *P(n,5) = 120*C(n,5)*, and he observed that *24 = 6*4*, and *120 = 24*5*. Based on these results, he then conjectured that "my next guess would be 120 times six, which would be 720," which is to say that he conjectured *P(n,6) = 720*C(n,6)*. This is correct, and in describing the pattern he states "What I noticed is each time you go up, you multiply by the next number. So 6 times 4 equals 24, which multiplied by 5 equals 120, which multiplied by 6 equals 720." To formalize his conjecture regarding the ratio between *P(n,k)* and *C(n,k)* for any parameter values *n* and *k*, Allen wrote the expression in Figure 2, which he characterized as his original solution, *P(n,k)*, divided by *k!*. Although he had found the correct closed-form expression for the binomial coefficient, he had done so solely through empirical patterning. The first author asked Allen why dividing by *k!* might yield the correct solution. Allen recognized that *k!* represents the number of ways to arrange *k* objects, which he used as justification for why dividing by *k!* yields the correct answer. We view this justification as a shift from RPG to PPG because Allen moved from using empirical results to create patterns, to justifying those patterns structurally by arguing how the mathematical expression connects to the outcomes he was counting. Hence, this episode demonstrates ER because it presents an instance of finding a pattern (a constant ratio between permutations and combinations), forming an associated generalization (the closed-form expression for the binomial coefficient), and re-interpreting the generalization from a structural perspective (justifying the ratio based on the outcomes being counted).

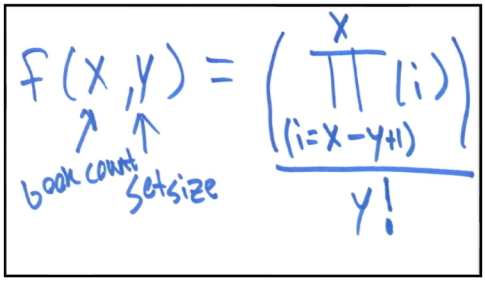

Figure 2: Allen's closed-form expression for the binomial coefficient

Having presented Allen's work, we now discuss three ways in which the computational environment facilitated the ER: (1) isolating generalizable relationships, (2) generating data, and (3) validating conjectures. Together these highlight specific ways in which the computational environment served to support this practice of ER.

Isolating Generalizable Relationships

Allen's generalizing activity in this TE centered on the relationship between *P(n,k)* and *C(n,k)*, which is the relationship between a mathematical expression he found by hand, and a numerical solution he found computationally. Because *C(n,k)* is a 2-variable function, isolating this relationship required understanding the relationship as both variables changed. In Allen's code, the variable *k* was hardcoded as the number of *for* loops, while the variable *n* was an assigned integer. Allen could easily change the value for *n*, which he used to his advantage by cycling through values of *n* in a *for* loop and observing the relationship between *P(n,k)* and *C(n,k)* for a fixed value *k*. However, changing the value for *k* required writing a new function, which can explain why Allen only examined the relationship for *k = 3, 4, 5, 6*. Having the two variable values manifested differently in the computer programs may have been beneficial, as it forced Allen to only change one variable value at a time. Creating conjectures that generalize the

effect of changing one value while keeping the other constant may have been easier than if he had the ability to change both variable values simultaneously.

Generating Data Sets

Allen wrote computer programs to generate sets of outcomes that are intractable to write by hand. To do so, he was required to engage with the sets themselves, as generating them required creating an algorithm that enumerated every outcome. His conjectures evolved as he generated each data set, which may indicate that writing the computer programs is more beneficial to ER than if he were given the data sets by the researchers. In addition, Allen was unsure how to calculate $C(n,k)$ without writing a computer program. Hence, we posit that this episode would not have occurred if not for the computational environment.

Validating Conjectures

Much of Allen's work iteratively moved between conjecturing and validating conjectures. The computational environment seemed to positively impact how he formed conjectures by motivating him to form conjectures that he could validate computationally. Every time Allen formed a conjecture, he immediately wrote a computer program to validate that conjecture. What Allen decided to conjecture about may have been informed by the elements of his computer program that seemed immediately adaptable. For example, Allen could easily change the value of n, and his initial conjectures formed around how n impacted the numerical solution. We posit that Allen's mastery of basic computer programming allowed flexibility in how he could adapt his code to accommodate changes in variable values. This flexibility manifested in his conjecturing by allowing some flexibility in the conjectures he was able to validate.

Discussion

In this paper, we have attempted to demonstrate a case in which a student leveraged a computational environment (and his computational facility) to engage with a combinatorial task, particularly using the computer meaningfully to engage in ER. We see much potential for ER to be a useful construct in considering students' mathematical activity (particularly relating to the practices of generalizing and conjecturing). In light of this, we think it is important to explore different aspects of this construct. Further, given the growing importance and ubiquity of computing in mathematics, and mathematics education, we see value in particularly exploring ways in which ER may be reinforced via the computer (and, reflexively, how a computational environment can specifically support practices such as ER). In this way, our findings contribute to a growing body of literature that demonstrates ways in which computational activity supports students' mathematical thinking and activity. Moreover, our findings also contribute to literature that demonstrates students using a computational environment in sophisticated ways that reflect the practices of research mathematicians.

We were fortunate in this study to uncover a case of a student leveraging the computer in a powerful way to support ER. Further research could more explicitly target not only instances of students using the computer meaningfully, but could investigate ways to support and engender productive use of the computer to support students' engagement in practices such as ER.

Acknowledgments

This research is supported by the National Science Foundation Grant 1650943.

References

Amit, M., & Neria, D. (2008). Rising to the challenge": Using generalization in pattern problems to unearth the algebraic skills of talented pre-algebra students. *ZDM Mathematics Education*, 40, 111–129.

Bills, L., & Rowland, T. (1999). Examples, generalisation and proof. *Research in Mathematics Education*, 1(1), 103–116.

Buteau, C., & Muller, E. (2017). Assessment in undergraduate programming-based mathematics courses. *Digital Experiences in Mathematics Education, 3,* 97–114. https://doi.org/10.1007/s40751-016-0026-4.

De Chenne, A., & Lockwood, E. (2022a). A task to connect counting processes to lists of outcomes in combinatorics. *Journal of Mathematical Behavior, 65*, DOI: 10.1016/j.jmathb.2021.100932

de Villiers, M. (2010). Experimentation and proof in mathematics. In G. Hanna, H.N. Jahnke, & H. Pulte (Eds.), Explanation and proof in mathematics (pp. 205-221). Springer, Boston, MA.

diSessa, A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education, *Mathematical Thinking and Learning, 20*(1), p. 3-31, DOI: 10.1080/10986065.2018.1403544

Ellis, A., Lockwood, E., & Ozaltun-Celik, A. (2022). Empirical re-conceptualization: From empirical generalizations to insight and understanding. *Journal of Mathematical Behavior, 65*(1). https://doi.org/10.1016/j.jmathb.2021.100928

Harel, G. (2001). The development of mathematical induction as a proof scheme: A Model for DNR-based instruction. In S. Campbell & R. Zaskis (Eds.), Learning and teaching number theory (pp. 185—212). Norwood, NJ: Ablex.

Harel, G., & Soto, O. (2017). Structural reasoning. International Journal of Research in Undergraduate Mathematics Education, 3(1), 225 – 242.

Knuth, E. J., Choppin, J., & Bieda, K. (2009). Middle school students' production of mathematical justifications. In D. A. Stylianou, M. L. Blanton, & E. J. Knuth (Eds.), Teaching and learning proof across the grades: A K-16 perspective (pp. 153-170). New York, NY: Routledge.

Küchemann, D. (2010). Using patterns generically to see structure. *Pedagogies: An International Journal, 5*(3), 233-250.

Lockwood, E., DeJarnette, A. F., & Thomas, M. (2019). Computing as a mathematical disciplinary practice. *Journal of Mathematical Behavior*, *54*(1). https://doi.org/10.1016/j.jmathb.2019.01.004.

Purdy, B., & Lockwood, E. (2020). An Example of Computational Thinking in Mathematics. In the Electronic Proceedings of the 23rd *Annual Meeting of the Research on Undergraduate Mathematics Education*. Boston, MA: Boston University.

Sand, O. P., Lockwood, E., Caballero, M., Mørken, K. (2022). Students' Development of a Logarithm Function in Python Using Taylor Expansions: a Teaching Design Case Study. *Digital Experiences in Mathematics Education, 8*(2), p. 213-255. https://doi.org/10.1007/s40751-022-00104-3

Steffe, L. P., & Thompson, P. W. (2000). Teaching experiment methodology: Underlying principles and essential elements. In R. Lesh & A. E. Kelly (Eds.), *Research design in mathematics and science education* (pp. 267–307). Mahwah: Lawrence Erlbaum Associates.

Stylianides, G., & Stylianides, J. (2009). Facilitating the transition from empirical arguments to proof. *Journal for Research in Mathematics Education*, *40*(3), 314–352.