Peer Tutoring vs. Solo Activities: Effects on Learning and Emotion

Ronessa Dass (ronessa.dass@carleton.ca)
Department of Cognitive Science, Carleton University, Ottawa, Canada

Kaleigh St Jacques (kaleigh.stjacques@carleton.ca)
Department of Cognitive Science, Carleton University, Ottawa, Canada

Kasia Muldner (kasia.muldner@carleton.ca)
Department of Cognitive Science, Carleton University, Ottawa, Canada

Abstract

Code tracing involves simulating at a high level the steps a computer takes when it executes a computer program. This is a fundamental skill needed for programming activities, but one that novices find challenging. Thus, work is needed on how to support novice programmers in this activity. We conducted an experimental study with university students (N = 56) learning to code trace in two conditions, namely peer tutored and solo code tracing. Our primary outcome variable was learning, but we also measured student emotions. Contrary to prior work in other domains, there was no significant benefit of peer tutoring and self-reported levels of emotion were similar in the two conditions; Bayesian statistics provided evidence for the null model in the majority of cases.

Introduction

Code tracing involves simulating at a high level the actions a computer takes when executing a computer program. This activity provides opportunities to learn important concepts, including: (1) the meaning of a programming language's syntax; (2) the rules of program execution. These concepts form the foundations needed for subsequent programming activities like code generation (Xie et al., 2019). Since learning to code trace is challenging (Cunningham, Blanchard, Ericson, & Guzdial, 2017; Fitzgerald, Simon, & Thomas, 2005; Vainio & Sajaniemi, 2007), effective teaching methods are needed. We next review research on helping students acquire this important skill.

Code tracing can be taught in a variety of ways, including live demonstrations in classrooms (Hertz & Jump, 2013), one-on-one tutoring (Xie, Nelson, & Ko, 2018), instructional videos (Lee & Muldner, 2020), and educational technologies like tutoring systems (Kumar, 2014; Nelson, Xie, & Ko, 2017; Jennings & Muldner, 2021; Caughey & Muldner, 2023). Code tracing can also be scaffolded with program visualization tools, which show the step-by-step flow of execution in a program and the program state at each step. In a recent review, Muldner et al. (2022) reported that visualization tools often, but not always, improved learning and/or performance over standard instruction without such tools.

Yet another method, which we focus on in the present paper, corresponds to peer tutoring. Roscoe and Chi (Roscoe & Chi, 2007) define peer tutoring as *"the recruitment of one student to provide one-on-one instruction for another student, accompanied by explicit assignment of participants to tutor and tutee roles"*. In this paradigm, the tutor and tutees are of similar age and are assigned specific roles (i.e., tutor, tutee), and either stay in those roles (fixed role) or take turns playing each role (reciprocal tutoring). While in some studies the peer doing the tutoring has more expertise, this is not always the case. Peer tutoring helps both the peer tutor and the tutee learn (Roscoe & Chi, 2007; Leung, 2019; Goodrich, 2018; Ansuategui & Miravet, 2017; Alegre, Moliner, Maroto, & Lorenzo-Valentin, 2019). This is encouraging given that peer tutors often have little or no tutoring expertise, and in the case of reciprocal tutoring, limited domain expertise.

Why is peer tutoring effective? To learn, students need opportunities to be constructively engaged with the materials (Chi & Wylie, 2014). Peer tutoring provides this opportunity for both the tutor and tutee. As far as the tutor, effective tutoring strategies include asking questions and providing knowledge-building explanations (for a review see (Roscoe & Chi, 2007)). Briefly, knowledge-building explanations require tutors to draw on their prior knowledge and to extend it through additional inferences, integration of ideas, and so on, which promotes their learning. This is in contrast to *knowledge telling* explanations, where the tutor lectures on what they already know with little reflection or extension. If tutors are using knowledge-building strategies and asking questions, then their tutees have many opportunities for their own knowledge construction.

We next describe work using peer tutoring and related paradigms in the domain of programming. Golding et al. (2005) conducted a study in the context of a first-year programming class over one semester, where class activities were done either alone or in a peer-tutoring context (N = 42; 60% male students). There was no significant difference between the conditions in terms of learning or student attitudes. Spacco et al. (2013) compared outcomes from two sections of a non-majors CS0 course taught by the same instructor. One section (N = 90) used a peer-instruction approach, where students did readings before class, wrote a brief quiz at the start of class, and then worked collaboratively on problems. This study is related to the peer tutoring paradigm but not identical, as the collaboration was not structured in the form of peer instruction. The other section (N = 124) was taught using a standard lecture-based approach. The peer instruction class performed significantly better on the final. Zingaro (2014) used the same design but with two different instructors teaching the two sections; self-efficacy was additionally measured

In L. K. Samuelson, S. L. Frank, M. Toneva, A. Mackey, & E. Hazeltine (Eds.), *Proceedings of the 46th Annual Conference of the Cognitive Science Society.*

(N = 221). There was no significant learning benefit of peer instruction but the students in the peer-instruction class had significantly higher self-efficacy.

The other studies we found did not use an experimental design. Gerhardt and Olan (2010) created a peer tutoring drop-in center. While many students did not take advantage of this service, those who did reported finding the tutoring useful and being satisfied with the tutoring. Crabtree et al. (2022) also recruited recent graduates of a programming course to act as peer mentors. The mentors attended lectures and answered student questions. There was no significant effect of peer tutors on class grades, as compared to a class that did not include them. Beyond peer-tutoring, a related paradigm is pair programming, where students collaborate to complete a programming task (but are not assigned specific tutor and tutee roles). In this context, reviews comparing pair programming to solo activities reported that pair programming improved outcomes like exam scores (Umapathy & Ritzhaupt, 2017).

Current Study

As our review highlights, there is ongoing research on the design of instructional materials and methods to support learning of code tracing. While peer tutoring is a promising approach that yields moderate to high effects over other forms of instruction, to date there is little research on its effect for programming activities, and none to the best of our knowledge that involves code-tracing activities.

Accordingly, the present study investigates the effect of peer tutoring on learning and emotion related to code-tracing activities. We included emotion in our analysis because it influences student learning (Pekrun, Elliot, & Maier, 2009; Kim & Pekrun, 2014; Camacho-Morles, Slemp, Oades, Morrish, & Scoular, 2019; Camacho-Morles et al., 2021). For instance, anxiety can reduce learning (Pekrun et al., 2009), while enjoyment can increase it (MacIntyre & Vincze, 2017). We compared peer tutoring outcomes to ones from individual code-tracing activities where students worked alone.

We had the following two research questions:

RQ1 Does peer tutoring increase learning of code tracing over solo activities?

RQ2 Does context (peer tutoring, solo activities) affect students' emotions?

Methods

Materials

Introductory Python Lesson We created a 22-minute instructional video to provide an introduction to code tracing with the language Python. The video featured a human narrator going over a PowerPoint slide deck that covered variables, assignment, if-else blocks, and while loops, as well as how to code trace programs.

Code-Tracing Problems and Solutions We designed four code-tracing problems based on the introductory lesson. All four problems showed a Python program with a *while* loop that contained an *if-else* block and a loop control variable (see Figure 1, left); the last two problems additionally included the accumulation of values in a second variable (see Figure 1, right) and thus were more challenging.

```
counter = 5
while True:
    counter = counter -1
    if counter < 3:
        print("ABC")
        break
    else:
        print("XYZ")
print("OK", counter)
```

```
counter = 0
res = 0
while True:
    counter = counter + 1
    if counter > 2:
        print("Hey")
        break
    else:
        res = res + counter
    print("Bye")
print(counter, res)
```

Figure 1: Example of two Python programs used in code-tracing activities.

We created four solutions sheets, one per problem. Each solution sheet showed a Python program on the left and a detailed step-by-step code-trace on the right. The code trace was presented in text form, as in some prior work (Bayman & Mayer, 1988). An alternative way to present a code trace is through an instructional video that shows, for instance, an instructor generating the code trace in real time (e.g., (Lee & Muldner, 2020)). We decided against this latter format because in one of our conditions, participants worked in pairs to discuss the solution, which would be challenging with a video (since the video would have to be paused each time discussion occurred).

Peer Tutoring Guidelines To inform participants in the peer tutoring condition on effective peer tutoring strategies, we created a brief lesson based on PowerPoint slide deck and a corresponding script read by a researcher. The lesson included information about the difference between knowledge telling and knowledge building and instructed participants to use the latter. The lesson also described other productive tutoring techniques, including provision of examples, question posing, and scaffolding of the interaction through prompts.

Pretest, Posttest, and Emotion Questionnaire A pretest and posttest were used to measure learning, each included eight code-tracing questions (five included while loops and three were more basic) and one code-writing transfer question. The tests were isomorphic (i.e., the questions were structurally identical but superficial features like variable names and numeric values were varied). The questions in the tests were similar but not identical to four code-tracing problems. An emotion questionnaire was used to measure five emotions, namely anxiety, boredom, enjoyment, frustration and confusion, on a Likert scale of 1 (not at all) to 5 (extremely). The five emotions were selected because they are relevant to academic settings (Pekrun et al., 2009). A separate Likert scale was used for each emotion.

Participants

The participants were 56 individuals (53 university students and three individuals not in university; M_{age}=21.2; 38 identified as female, 15 as male, 1 as non-binary, 1 as demigirl, and two preferred not to answer). To be eligible, participants needed to have either no programming experience or limited experience (no more than one university course), in order to avoid ceiling effects on the tests. The majority of participants had no prior programming experience (53.6%); the other participants had limited experience (high school or at most one university course). Participants were recruited through word of mouth, several social media groups on Facebook, and through SONA (an online participant management tool). The SONA participants were all enrolled in a first year cognitive science class and received 2% course credit; the other participants received $25 compensation.

Design and Procedure

The study was approved by the University Ethics Board. A between-subjects design was used with two conditions: peer tutoring (participants worked on code-tracing problems in pairs, taking turns to tutor one another) and solo (participants worked alone). Participants were assigned to the conditions in a round robin fashion. Participants in the peer tutoring condition were required to come to the study with a friend.

Each study session was conducted over Zoom. Participants first signed an informed consent form. The researcher then used screen sharing to show the Python lesson. Next, participants were given a maximum of fifteen minutes to complete the pretest. Participants were instructed to not guess their answers and to display all of their work. The test was administered after the lesson because we wanted to isolate any effects of peer tutoring to the intervention (rather than inflating effects with the influence of the lesson). After the pretest, participants worked on the four code-tracing problems. Participants were told they could refer to the Python lesson if they wished and were given a help sheet corresponding to a slide from the lesson showing a detailed code trace of a program (this problem was similar to but not identical to the four problems, in that it used a while loop but the order of constructs in the loop body was different).

In the peer tutoring condition, participants worked in pairs and took turns playing the role of the tutor and the tutee. One partner (partner A) played the role of the tutor for code-tracing problems (1) and (4) and the second partner (partner B) played the role of the tutor for problems (2) and (3). Recall that problems 1 and 2 were simpler than problems 3 and 4. This sequencing ensured that each participant tutored with an easy and harder problem and was also tutored with an easier and harder problem. In sum, reciprocal tutoring was used and both participants had the opportunity to be a tutor and a tutee. When playing the role of tutee, participants worked on the given problem, while their tutor answered questions and provided support, as needed; these roles flipped on the next problem. Participants were required to spend a minimum of three minutes per problem and were given a maximum of seven minutes per problem. After completion of a problem, or after the allotted time had passed, participants were given the solution sheet (via a link in Zoom chat) and had three minutes to review and/or revise their solution.

The procedure in the solo condition was the same, except that participants worked alone and were asked to think aloud (Ericsson & Simon, 1980) while working on the four problems. Thinking aloud involves verbalizing the contents of working memory (without further prompts to explain verbalizations). Before participants begun working on the code-tracing problems, a researcher described what thinking aloud involved using a predefined script. We asked the solo participants to think aloud to encourage engagement with the content and to increase consistency between conditions, given that participants in the peer tutoring condition also verbalized ideas. Solo participants were given a maximum of seven minutes per problem and had to spend a minimum of two minutes per problem. A minimum of two minutes was used because no dialogue was required in this condition, which reduced time on task, and requiring participants to keep talking could induce frustration and/or superficial comments. After participants stated that were finished the problem, or after seven minutes had passed, they were given the solution sheet and had three minutes to review and/or revise their solution.

In both conditions, after the pretest and after each code-tracing problem, participants completed the emotion self-report questionnaire. After the final code-tracing problem and self-report questionnaire, participants individually completed the posttest (20 minutes). Participants were asked not to guess and to show their work. After the posttest participants were debriefed and compensated. The entire study, in both conditions, took no more than two hours.

Results

For the analysis, we used both frequentist statistics (null hypothesis significance testing, NHST) and Bayesian statistics (Bayes Factor, BF). In the latter, the *"likelihood of the data is considered under both the null and alternative hypotheses, and these probabilities are compared via the Bayes factor. The Bayes factor is a ratio that contrasts the likelihood of the data fitting under the null hypothesis with the likelihood of fitting under the alternative hypothesis"* (Jarosz & Wiley, 2014). The ratio can be computed in either direction (i.e., to show the results from the perspective of the alternative hypothesis or the null hypothesis). A key advantage of this method is that it can provide evidence for either model, null or alternative (as opposed to NHST that can only provide evidence for rejecting the null model). Thus, there have been calls to present results from both frameworks (NHST, Bayesian), so that complimentary evidence can be compared (Jarosz & Wiley, 2014; Quintana & Williams, 2018).

As is standard, we report the Bayes factor for the more likely model: BF_{01} when the null model is more likely (no conditional difference) and BF_{10} when the alternative model

Table 1: Descriptives for the pretest, posttest, and gain scores for each condition

	Peer Tutoring $n = 26$	Solo $n = 30$
Pretest %	34.07% (21.13)	36.11% (21.43)
Posttest %	55.77% (24.30)	62.70% (22.94)
Gain %	21.70% (19.33)	26.75% (18.40)

is more likely (conditional difference exists) as appropriate, stating which direction we are reporting. Of note, the other BF factor can be calculated simply by inverting the reported one. When either Bayes factor is close to 1, this indicates lack of evidence for either model being superior. As the Bayes factor increases, it provides mounting evidence for the target model (either null or alternative, depending on the way the ratio is set up). We follow the guidelines in (Jarosz & Wiley, 2014) to interpret Bayes factors, as follows: BF = 1-3 provides anecdotal evidence for the corresponding model; BF = 3 - 10 provides substantial evidence for the corresponding model; and $BF > 100$ provides decisive evidence. To obtain the Bayes factor, we used JASP (JASP Team, 2019).

Learning Results

Learning was measured using the difference from pretest to posttest. The tests were graded using a detailed grading scheme, blind to condition. As shown in Table 1, the pretest scores were low and similar in both conditions, $t(54) = .36$, $p = .72$, $d = .01$; $BF_{01} = 3.50$ indicated substantial evidence for the null model. No participant was at ceiling, as all pretest scores were below 80%.

Ignoring condition, participants did learn from the instructional activities as indicated by the significant gain from pretest to posttest, $t(51) = 10.60$, $p < .001$, $d = 1.47$; $BF_{10} > 100$ indicated decisive evidence for the alternative model. The effect of condition on learning was not significant, $t(54) = 1.00$, $p = .32$, $d = .27$; $BF_{01} = 2.44$ indicated anecdotal evidence for the null model. Four participants did not learn (either had no gain or reduced performance at posttest; three were in the peer tutoring condition). Lack of learning can be the result of either students rushing through the last activity, namely the posttest, or the instructional materials causing confusion. Since only four out of 56 participants did not learn in the current study, the latter option is unlikely and the more plausible explanation is that these participants did not invest effort in the posttest. If we re-run the learning results without these participants, the pattern for frequentist statistics remains ($p > .05$) but the Bayes factor now provides substantial evidence for the null model indicating lack of a peer-tutoring benefit ($BF_{01} = 3.14$).

Emotion Results

We now turn to the results related to the five emotions measured in our study (anxiety, boredom, confusion, enjoyment, frustration). Recall participants were prompted to self-report on their emotions at five points, namely after the pretest and each of the four code-tracing problems. We asked for emotion information after each problem rather than at the end of the experimental session to avoid losing information, which would happen if emotions fluctuated during the study.

We begin with the descriptives, shown in Figure 2. We focus on the last four self-reports produced after each code-tracing problem (T2-T5 in Figure 2). In the majority of cases for each of the five emotions, the line in Figure 2 for each condition tends to be fairly flat, indicating little difference between the emotion levels at the four time points within that condition. Similar trends appeared in both conditions. Anxiety started off moderately low (< 2.5 out of 5), eventually dropping slightly to 2 by the last problem. Boredom stayed low and constant (around 2 out of 5), which makes sense given that participants had to engage in a problem-solving activity (and so were active, which likely reduced boredom). Confusion followed a similar trend as boredom. The relatively low levels of confusion were initially surprising as we recruited novices but this state may have been mitigated by the feedback provided by the solution sheets. Enjoyment was moderate throughout (around 3 out of 5); compared to the similar levels reported in the solo condition, the peer tutoring group's enjoyment increased slightly by the fourth problem (around 3.5 out of 5). Frustration was low and similar in the two conditions with one exception, namely after the second code-tracing problem. Here, the solo group's frustration increased while the peer-tutoring group's frustration decreased, compared to the amount reported after problem 1.

Given the fact that there was little variation in emotion level between the four key time points (i.e., after each code-tracing activity) and given this was the case in each condition, the most straightforward way to analyze the data to obtain the effect of condition is as follows: (1) compute the mean self-reported level of each emotion across the time points (focusing on the reports after each problem), and (2) compare the result between the two conditions using a separate independent t-test for each emotion (frequentist, Bayesian). The results are in Table 2. In sum, there was no significant difference between the conditions in terms of emotion level reported and the effect sizes were all low except for frustration. The Bayesian analysis reported substantial evidence for the null model for anxiety, boredom, confusion (i.e., no conditional effect on these emotions), while for enjoyment, the evidence for the null model was only anecdotal. For frustration both the null and alternative models were equally likely, so no conclusions can be drawn about the effect of condition on frustration.

The alternative way to analyze the data is to run mixed two-way ANOVAs with condition as the between-subjects factor (solo, peer tutoring) and time point as the four-level within

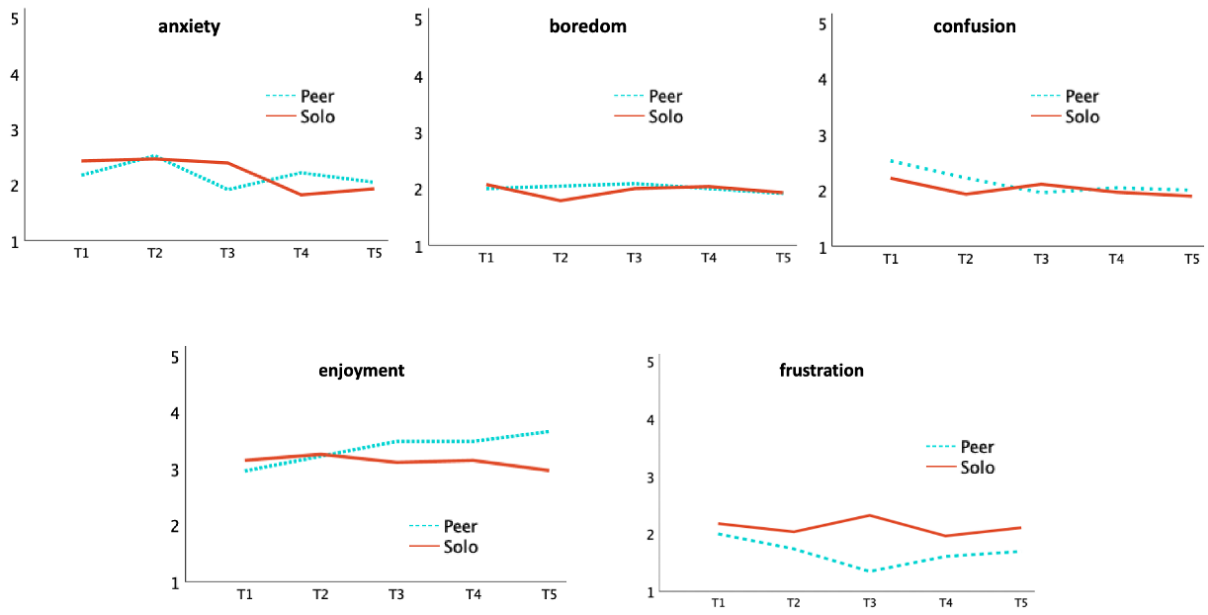

Figure 2: Mean self-reported emotion levels for the 5 target emotions in each condition (solo and peer tutoring). Y-axis shows mean reported level for the corresponding emotion (1 indicating not all feeling that emotion and 5 indicating extremely); X-axis shows the five time points at which the self-report was administered: T1 = right after the pretest, and T1-T4 right after the corresponding code-trace problem.

Table 2: Frequentist and Bayesian inferential statistics for the effect of condition on mean self-reported emotions.

	Frequentist Statistics (NHST)	Bayes Factor (BF)
anxiety	$t = 0.1$, $p =. 93$, $d = .03$	$BF_{01} = 3.6$
boredom	$t = 0.4$, $p = .71$, $d = .10$	$BF_{01} = 3.4$
confusion	$t = 0.2$, $p = .88$, $d = .04$	$BF_{01} = 3.5$
enjoyment	$t = 1.3$, $p = .23$, $d = .24$	$BF_{01} = 2.0$
frustration	$t = 1.7$, $p = .09$, $d = .49$	$BF_{01} = 1.0$

subject factor (focusing on the self-reports after each code-tracing activity). This analysis makes it possible to check for trends over time. Based on the descriptives, the only potential trend is for frustration and this indeed turned out to be the case. Specifically, the effect of condition on self-reported level of anxiety, boredom, confusion, and enjoyment was not significant and neither was the condition x time interaction (all $p < 0.05$); these results were mirrored with decisive evidence for the null model produced by the Bayesian analysis. This is evident from Figure 2, given the relatively flat lines, indicating little change over time, and the relatively parallel slopes in terms of the two conditions, indicating lack of interaction. The one exception was for frustration. Specifically, as noted above, frustration spiked after the second problem in the solo condition but decreased slightly in the peer condition. Frequentist statistics confirmed this pattern through a significant condition x time interaction, $F(3, 147) = 2.93$, $p = .036$, $\eta_p^2 = .056$, likely driven by the difference between the conditions after the third self-report right after the first complex problem was solved. However, these results have to be interpreted with caution, as there is virtually no evidence for the effect of this interaction through the Bayesian statistics - the inclusion *BF* used to interpret results from mixed ANOVAs (JASP, 2023) is .74, which indicates that the null and alternative models are similarly likely.

Discussion

We conducted an experimental study that involved code-tracing activities ($N = 56$). In the peer tutoring condition, students engaged in reciprocal tutoring, while in the solo condition students worked alone. There was no significant effect of condition on learning (RQ1) and the effect size was small for all analyses. Bayesian statistics provided evidence for the null model (no conditional effect), trending towards substantial with all participants and substantial when four participants who did not learn were excluded. There was also no significant effect of learning context on emotion (RQ2), with the possible exception of frustration.

Meta-analysis in domains outside of programming activities indicated peer tutoring resulted in more learning than other activities, like solo problem solving. Why was this not the case in our study? The posttest scores were not at ceil-

ing and in general had room for improvement (below 65%); students did learn overall, so these are not potential explanations. Neither is the fact that the solo condition was asked to verbalize their thoughts, because they were not asked to self-explain their reasoning. Had the latter been the case, then this would have boosted the solo condition's performance as self-explanation improves learning (Chi & Wylie, 2014).

We recruited novice tutors. Prior work shows that even novice tutors can help their tutees learn (Robinson, Schofield, & Steers-Wentzell1, 2005), but it is possible our tutors did not do enough prompting (good for learning and did too much telling (bad for learning), which would have reduced learning in the peer-tutoring condition. To check for this possibility, we analyzed a subset of the transcripts corresponding to six of the peer tutoring sessions (each one involved a pair or participants, for a total of $n = 12$). Prompting was defined as guiding the tutee towards the answer without giving the full answer away via hints, similar examples, open-ended prompts, while telling corresponded to the tutor telling the answer[1].

The peer tutors in our study did slightly more prompting than telling (on average per problem, 1.83 vs. 1.41, respectively). Examples of prompting included: *"when we looked at the example, where was the variable that we were talking about?"* and "*okay, how do you want to go about this? What are we setting the counter?"*. In these examples, the tutor is guiding the tutee to the solution by reminding them of past examples and by having them reflect on the problem. Examples of telling included: *"since line 10 is still within the indented loop for the while loop, I think we print out bye"* and *"yeah and then you go back to line seven"*. Here, the tutor is telling the tutee the answer, instead of helping their tutee work construct it. The fact there was more prompting than telling is encouraging because prompting is beneficial for tutor and tutee learning. Thus, this aspect is not a likely cause of the lack of a peer tutoring benefit.

Yet another possibility to explain lack of a peer tutoring benefit relates to the fact that in our study, participants were given the canonical solution after each problem in both conditions. Feedback is highly beneficial, with a slightly larger effect size than peer tutoring, .62 vs. .51 (Hattie, John, 2023). Thus, the provision of feedback may have overshadowed any peer tutoring effects.

We now turn to the emotion results. Peer tutoring involves collaborating with another individual. In general, the quality of collaboration influences emotions experienced (Pietarinen, Vauras, Laakkonen, Kinnunen, & Volet, 2019; Zschocke, Wosnitza, & Bürger, 2016).[2] Since we recruited participants who knew each other, we expected this would have boosted the collaboration and thus increased positive emotions over solo activities. We did not, however, find evidence of this. The only significant effect corresponded to frustration, which increased after code-tracing problem 2 in the solo condition and decreased in the peer tutoring condition. It's not clear why frustration would be affected by problem 2 in this way, particularly since the levels of frustration at other points were similar. Since Bayesian statistics did not provide reliable evidence for this effect, it must be interpreted with caution.

Limitations and Future Work

The study was conducted over Zoom rather than as part of regular class activities, which may have influenced motivation to complete the activities. Conducting a study in the context of a regular class increases ecological validity but reduces experimental control. We recruited participants for the peer tutoring condition who already knew each other. This method may help participants be more comfortable exchanging ideas but it also means that our results may not generalize to settings where the tutor and tutee do not know each other.

The modest sample size in the present study influences the power of frequentist statistics. To mitigate this limitation, we followed the advocated practice of reporting effect sizes, which are informative in modest power situations. We also used Bayesian statistics that are not as influenced by sample size (Dienes, 2014).

We measured emotion using a questionnaire. While it was administered several times, it was brief (five questions) and so unlikely to be disruptive. However, future work could investigate alternative approaches that don't involve asking participants to self-report. In our prior work we analyzed the talk aloud data for emotional content (Savelson & Muldner, 2023). This approach has the advantage of not asking participants directly how they are feeling, but it also makes it more challenging to identify certain emotions because participants do not express them much (or at all). Future work should also investigate the influence of the activities students are asked to complete. Here, code tracing was used, which is a procedural task that affords limited opportunity for elaboration. It may be that for peer tutoring to have an effect, a richer problem is needed, like program generation.

Conclusion

Meta-reviews of peer tutoring show it increases learning over other instructional activities like working alone. However, very little work exists in this paradigm involving programming activities. Although we did not find evidence that peer tutoring increased learning or influenced emotions over solo activities related to code tracing, it is premature to conclude that peer tutoring is not effective for programming activities. Thus, further studies are needed, as well as research to identify factors that influence peer tutoring effectiveness in this domain.

Acknowledgements

This work was supported by an NSERC Discovery grant.

[1] Initially, four transcripts were coded by two researchers, and any disagreements were discussed; the coding scheme was refined to clarify as needed. The remaining transcripts were coded by one researcher.

[2] While the aforementioned research did not structure collaboration using peer tutoring roles, the results should still transfer to peer tutoring contexts.

References

Alegre, F., Moliner, L., Maroto, A., & Lorenzo-Valentin, G. (2019). Peer tutoring in mathematics in primary education: a systematic review. *Educational Review*, *71*(6), 767-791. doi: 10.1080/00131911.2018.1474176

Ansuategui, F. J. A., & Miravet, L. M. (2017). Emotional and cognitive effects of peer tutoring among secondary school mathematics students. *International Journal of Mathematical Education in Science and Technology*, *48*(8), 1185-1205. doi: 10.1080/0020739X.2017.1342284

Bayman, P., & Mayer, R. E. (1988). Using conceptual models to teach basic computer programming. *Journal of Educational Psychology*, *80*(3), 291. doi: https://doi.org/10.1037/0022-0663.80.3.291

Camacho-Morles, J., Slemp, G. R., Oades, L. G., Morrish, L., & Scoular, C. (2019). The role of achievement emotions in the collaborative problem-solving performance of adolescents. *Learning and Individual Differences*, *70*, 169–181.

Camacho-Morles, J., Slemp, G. R., Pekrun, R., Loderer, K., Hou, H., & Oades, L. G. (2021). Activity achievement emotions and academic performance: A meta-analysis. *Educational Psychology Review*, *33*, 1051 - 1095.

Caughey, M., & Muldner, K. (2023). Investigating the utility of self-explanation through translation activities with a code-tracing tutor. *International Conference on Artificial Intelligence in Education*, 66–77.

Chi, M., & Wylie, R. (2014). The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes. *Educational Psychologist*, *49*(4), 219–243.

Cunningham, K., Blanchard, S., Ericson, B., & Guzdial, M. (2017). Using tracing and sketching to solve programming problems: Replicating and extending an analysis of what students draw. In *Proceedings of the 2017 ACM conference on international computing education research* (p. 164–172). doi: 10.1145/3105726.3106190

Dienes, Z. (2014). Using bayes to get the most out of non-significant results. *Frontiers in Psychology*, *5*.

Ericsson, K. A., & Simon, H. A. (1980). Verbal reports as data. *Psychological review*, *87*(3), 215-251.

Facey-Shaw, L., & Golding, P. (2005). Effects of peer tutoring and attitude on academic performance of first year introductory programming students. In *Proceedings of Frontiers in education 35th annual conference.* doi: 10.1109/FIE.2005.1612175

Fitzgerald, S., Simon, B., & Thomas, L. (2005). Strategies that students use to trace code: an analysis based in grounded theory. In *Proceedings of the first international workshop on computing education research* (p. 69–80). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/1089786.1089793

Gerhardt, J., & Olan, M. (2010). Peer tutoring in programming: Lessons learned. *Information Systems Education Journal*, *8*(39).

Gerhardt, J., & Olan, M. (2022). Utilizing peer tutors in introductory programming education: An exploratory investigation. *International Journal of Teaching & Learning in Higher Education*, *33*, 429-445.

Goodrich, A. (2018). Peer mentoring and peer tutoring among k–12 students: A literature review. *Applications of Research in Music Education*, *36*(2), 13-21. doi: 10.1177/8755123317708765

Hattie, John. (2023). *Hattie's table of effect sizes.* Retrieved from `https://www.visiblelearningmetax.com/`

Hertz, M., & Jump, M. (2013). Trace-based teaching in early programming courses. In *Proceeding of the 44th acm technical symposium on computer science education* (p. 561–566). New York, NY, USA. doi: 10.1145/2445196.2445364

Jarosz, A. F., & Wiley, J. (2014). What are the odds? A practical guide to computing and reporting Bayes factors. *Journal of Problem Solving*, *7*, 2-9. doi: 10.7771/1932-6246.1167

JASP. (2023). *Bayesian guide v0.12.2.*

JASP Team. (2019). *JASP (Version 0.10.2)[Computer software].*

Jennings, J., & Muldner, K. (2021). When does scaffolding provide too much assistance? a code-tracing tutor investigation. *International Journal of Artificial Intelligence in Education*, *31*(4), 784–819. doi: https://doi.org/10.1007/s40593-020-00217-z

Kim, C., & Pekrun, R. (2014). Emotions and motivation in learning and performance. In J. M. Spector, M. D. Merrill, J. Elen, & M. J. Bishop (Eds.), *Handbook of research on educational communications and technology* (pp. 65–75). Springer New York.

Kumar, A. N. (2014). An Evaluation of Self-Explanation in a Programming Tutor. In *Proceedings of the 12th International Conference on Intelligent Tutoring Systems* (pp. 248–253). Berlin, Heidelberg: Springer-Verlag.

Lee, B., & Muldner, K. (2020). Instructional video design: Investigating the impact of monologue-and dialogue-style presentations. In *Proceedings of the 2020 CHI conference on human factors in computing systems* (pp. 1–12). New York, NY, USA. doi: https://doi.org/10.1145/3313831.3376845

Leung, K. C. (2019). An updated meta-analysis on the effect of peer tutoring on tutors' achievement. *School Psychology International*, *40*(2), 200-214. doi: 10.1177/0143034318808832

MacIntyre, P. D., & Vincze, L. (2017). Positive and negative emotions underlie motivation for l2 learning. *Studies in Second Language Learning and Teaching*, *7*(1), 61–88. doi: https://doi.org/10.14746/ssllt.2017.7.1.4

Muldner, K., Jennings, J., & Chiarelli, V. (2022, dec). A review of worked examples in programming activities. *ACM Transactions of Computing Education*, *23*(1). doi: 10.1145/3560266

Nelson, G. L., Xie, B., & Ko, A. J. (2017). Comprehension first: Evaluating a novel pedagogy and tutoring system for program tracing in CS1. In *Proceedings of the 2017 ACM*

conference on international computing education research (p. 2–11). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3105726.3106178

Pekrun, R., Elliot, A. J., & Maier, M. A. (2009). Achievement goals and achievement emotions: Testing a model of their joint relations with academic performance. *Journal of Educational Psychology*, *101*(1), 115–135. doi: 10.1037/a0013383

Pietarinen, T., Vauras, M., Laakkonen, E., Kinnunen, R., & Volet, S. (2019). High school students' perceptions of affect and collaboration during virtual science inquiry learning. *Journal of Computer Assisted Learning*, *35*(3), 334–348. doi: https://doi.org/10.1111/jcal.12334

Quintana, D. S., & Williams, D. R. (2018, Jun 07). Bayesian alternatives for common null-hypothesis significance tests in psychiatry: a non-technical guide using jasp. *BMC Psychiatry*, *18*(1), 178.

Robinson, D. R., Schofield, J. W., & Steers-Wentzell1, K. L. (2005). Peer and cross-age tutoring in math:outcomes and their design implications. *Educational Psychology Review*, *17*, 327-362.

Roscoe, R., & Chi, M. (2007, December). Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions. *Review of Educational Research*, *77*(4), 534–574. doi: 10.3102/0034654307309920

Savelson, Z., & Muldner, K. (2023). How do students feel and collaborate during programming activities in the productive failure paradigm. *Computer Science Education*, 1–34.

Simon, B., Parris, J., & Spacco, J. (2013). How we teach impacts student learning: Peer instruction vs. lecture in cs0. In *Proceeding of the 44th acm technical symposium on computer science education* (p. 41–46). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2445196.2445215

Umapathy, K., & Ritzhaupt, A. D. (2017, aug). A meta-analysis of pair-programming in computer programming courses: Implications for educational practice. *ACM Transactions of Computing Education*, *17*(4). doi: 10.1145/2996201

Vainio, V., & Sajaniemi, J. (2007). Factors in novice programmers' poor tracing skills. In *Proceedings of the 12th annual sigcse conference on innovation and technology in computer science education* (p. 236–240). New York, NY, USA. doi: 10.1145/1268784.1268853

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., … Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, *29*(2-3), 205-253. doi: 10.1080/08993408.2019.1565235

Xie, B., Nelson, G. L., & Ko, A. J. (2018). An explicit strategy to scaffold novice program tracing. In *Proceedings of the 49th acm technical symposium on computer science education* (p. 344–349). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3159450.3159527

Zingaro, D. (2014). Peer instruction contributes to self-efficacy in cs1. In *Proceedings of the 45th acm technical symposium on computer science education* (p. 373–378). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2538862.2538878

Zschocke, K., Wosnitza, M., & Bürger, K. (2016). Emotions in group work: insights from an appraisal-oriented perspective. *European Journal of Psychology of Education*, *31*(3), 359–384.