Pair Programming in Practice: A PhD Tutor's Reflections

Sarah Elliot

Usher Institute, University of Edinburgh
s.s.elliot@sms.ed.ac.uk

From Student to Teacher

I am a PhD candidate in healthcare data science, having learnt R and Python programming at university. I vividly remember my own experiences as a beginner: persistently feeling lost or behind my peers due to my non-coding background, jargon-filled error messages, and overwhelming online solutions. Alongside these frustrations, I also recall the satisfaction of mastering a new skill and the sense of achievement that followed solving a difficult problem. These memories shaped my approach to tutoring during my PhD. I was motivated to support students in acquiring technical skills, and, importantly, build confidence while stepping into a new programming space that can often feel intimidating.

I worked as a tutor on an introductory R course designed for beginners. Students enrolled with a wide range of experience levels: those with no technical background, individuals transitioning to R from another programming language, and those with some self-taught experience seeking a more structured workflow. The course was delivered online on a part-time basis, with most students working full-time as well as studying. This course format increased the accessibility for learners who might otherwise be excluded from traditional full-time study, but it also presented pedagogical challenges. Students were geographically dispersed, had additional personal and professional commitments, and had different technical home set-ups. Learning primarily took place through scheduled online sessions, online discussion boards and independent work, and opportunities for informal peer interactions were more limited. Therefore, the course coordinators utilised learning activities that supported both technical development and social connection — notably, pair programming.

Pair Programming Builds Confidence

The emergence of generative artificial intelligence (AI) has reshaped how beginner programmers can approach coding tasks. **Problems that previously required trial-and-error troubleshooting can be resolved within seconds of requesting an AI solution.** These AI tools are valuable but can reduce the understanding of the resulting code. Learning to program involves more than obtaining functional code: it is important to understand core concepts such as data and its structure, functions, good coding practices, problem solving, and transparent interpretation and communication of results. The best way to develop these skills is through actively engaging with problems as opposed to passively receiving an answer (Bonwell and Eison 1991; Chattopadhyay, Ryan, and Pockrandt 2022).

Pair programming is a strategy that prioritises learning the process of coding as opposed to just the end product. It involves at least two individuals working together on the same task at the same time, with one 'driver' who types the code and thinks out loud and one 'navigator' who instructs the driver, reflects on the code, and looks out for mistakes (Bryant, Romero, and Boulay 2008). Working collaboratively, students articulate their reasoning, suggest alternative approaches and interpret challenges together.

The confidence-building effects of this approach were evident across the students. More experienced students derived satisfaction from supporting their peers and consolidating their own understanding, while less experienced students gained practical experience in a low-risk setting. It is possible that students felt less anxious contributing ideas in small group settings as opposed to whole-class discussions (McDonough 2004). Observing the learners' engagement over time, the students were seen to become more confident undertaking both roles, with

less hesitation in sharing screens and contributing ideas. Coding alongside peers who were also in the process of developing their skills was valuable as explanations were drawn from recently experienced challenges and solutions, as opposed to second-nature expertise from a formal teacher. Challenges were faced with collective resilience as opposed to feeling like an individual struggle. Pairs actively gave and received feedback, providing learning opportunities that are missed during individual tasks (Storch and Wigglesworth 2006). Content and skills learned collaboratively, and practised during pair programming, seemed to be retained for longer and utilised in future work and assessments (Shehadeh 2011).

Due to having a relatively small cohort and strong staff-to-student ratio, the progress of individual students was visible to the teaching team during this course. I observed students who regularly engaged with pair programming benefit from confident engagement with the course overall. The hands-on programming gave them more coding practice than teacher-fronted activities, and the growing confidence in their community and their abilities resulted in willingness to ask questions and share partial solutions on the online discussion boards. Collaboratively solving challenges, communicating throughout learning, and becoming confident with uncertainty emerged as valuable skills developed from pair programming.

There was also a social benefit to pair programming. Due to running online, many students had few opportunities to interact with their peers outside of the scheduled sessions. Reflections from students highlighted that paired or grouped activities created opportunities for connection, helping students feel less isolated. Pair programming supported learning and community building, which can be difficult to achieve in a remote learning environment (Laal and Ghodsi 2012).

Supporting Each Student To Make Mixed Pairings Work

Implementing pair programming was not without difficulties. Grouping students with widely varying experience levels required close consideration. **Mixed-ability pairings are productive, but it is important to monitor the contributions of each student.** The experience gap should support collaborative learning and skill development, particularly among those paired with a more-proficient peer, without the less-experienced participant becoming too intimidated to contribute or the more-experienced participant feeling insufficiently challenged (Shehadeh 2011; Maguire et al. 2014).

Technical constraints also impacted participation. Due to the remote delivery and global reach of this course, it was common for some learners to have unstable internet connectivity or other technical issues that limited their ability to share their screen and take on the 'driver' role. This influenced the roles that individuals could choose and the group dynamics. This was improved by using groups of three where needed, allowing more flexibility in chosen roles.

Although collaborative learning is a valuable skill, it can be undermined if students see their classmates as competitors (Schinske and Tanner 2014). In this course, students are explicitly told that their final grades would not be in competition with their peers, and this framing helped encourage collaboration throughout pair programming. Throughout the pair coding tasks, students shared ideas, experiences, useful tools and innovative functions, which appeared to improve the quality of the learning and understanding of key concepts across the cohort.

A Richer Learning Experience for Both Students and Tutors

While facilitating pair programming, a tutor's role is to support students as they work through tasks and navigate challenges. This involves allowing students to attempt challenges before offering help and, when needed, providing helpful hints as opposed to full solutions, balancing guidance with autonomy. The nature of pair programming means students must verbalise their thought process, making it easier for tutors to assess the learners' needs and identify any misunderstandings. While the class is working in pairs, tutors have the opportunity to assist individual learners or pairs (Soleimani, Modirkhamene, and Sadeghi 2017). As more experienced R programmers, tutors can share practical, real-world approaches for students to explore. Tutors are able to observe pair dynamics to learn which pairs work well and who needs more support, to inform future groupings. Tutors should lead by example to support student learning: this includes communicating ideas clearly, modelling good coding practices, maintaining approachability (such as keeping cameras on in remote sessions), and asking reflective questions.

Reflecting on this experience, using pair programming to teach non-programmers resulted in a way of both building confidence in technical skills and fostering community in a remote environment. In an era where AI can readily generate solutions, pair programming allowed students to gain hands-on coding experience and programmatic thinking, while also developing transferable skills such as problem solving, communication, and collaborative learning. **As a PhD tutor, pair programming allowed me to share my own strategies during these sessions and also learn new approaches from the students, showing that learning is richest when multiple perspectives are shared at every level.**

References

Bonwell, Charles C., and James A. Eison. 1991. “Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports.” ERIC Clearinghouse on Higher Education, The George Washington University, One Dupont Circle, Suite 630, Washington, DC 20036-1183. https://eric.ed.gov/?id=ED336049.

Bryant, Sallyann, Pablo Romero, and Benedict du Boulay. 2008. “Pair Programming and the Mysterious Role of the Navigator.” *International Journal of Human-Computer Studies*, Collaborative and social aspects of computer software development, 66 (7): 519–29. https://doi.org/10.1016/j.ijhcs.2007.03.005.

Chattopadhyay, Ankur, Drew Ryan, and James Pockrandt. 2022. “Scaffolded Live Coding: A Hybrid Pedagogical Approach for Enhanced Teaching of Coding Skills.” In *2022 IEEE Frontiers in Education Conference (FIE)*, 1–9. IEEE. https://doi.org/10.1109/FIE56618.2022.9962513.

Laal, Marjan, and Seyed Mohammad Ghodsi. 2012. “Benefits of Collaborative Learning.” *Procedia - Social and Behavioral Sciences*, World conference on learning, teaching & administration - 2011, 31 (January): 486–90. https://doi.org/10.1016/j.sbspro.2011.12.091.

Maguire, Phil, Rebecca Maguire, Philip Hyland, and Patrick Marshall. 2014. “Enhancing Collaborative Learning Using Pair Programming: Who Benefits?” *All Ireland Journal of Higher Education* 6 (2). https://doi.org/10.62707/aishej.v6i2.141.

McDonough, Kim. 2004. “Learner-Learner Interaction During Pair and Small Group Activities in a Thai EFL Context.” *System: An International Journal of Educational Technology and Applied Linguistics* 32 (2): 207–24. https://doi.org/10.1016/j.system.2004.01.003.

Schinske, Jeffrey, and Kimberly Tanner. 2014. “Teaching More by Grading Less (or Differently).” *CBE Life Sciences Education* 13 (2): 159–66. https://doi.org/10.1187/cbe.cbe-14-03-0054.

Shehadeh, Ali. 2011. “Effects and Student Perceptions of Collaborative Writing in L2.” *Journal of Second Language Writing* 20 (4): 286–305. https://doi.org/10.1016/j.jslw.2011.05.010.

Soleimani, Maryam, Sima Modirkhamene, and Karim Sadeghi. 2017. “Peer-Mediated Vs. Individual Writing: Measuring Fluency, Complexity, and Accuracy in Writing.” *Innovation in Language Learning and Teaching* 11 (1): 86–100. https://doi.org/10.1080/17501229.2015.1043915.

Storch, Neomy, and Gillian Wigglesworth. 2006. “Writing Tasks:: The Effects of Collaboration.” In *Investigating Tasks in Formal Language Learning*, 157–77. Multilingual Matters. https://doi.org/10.21832/9781853599286-011.