Sequential vs. Simultaneous: Approaches to Learning Programming and Statistics

Robert S. Young

Edinburgh Medical School, University of Edinburgh; Zhejiang University - University of Edinburgh (ZJE) Institute
robert.young@ed.ac.uk

Rebecca L. Colquhoun

Department of Earth Science and Engineering, Imperial College London
r.colquhoun23@imperial.ac.uk

Tiago A. Marques

School of Mathematics, University of St Andrews
tiago.marques@st-andrews.ac.uk

Brittany Blankinship

Edinburgh Medical School, University of Edinburgh
B.Blankinship@ed.ac.uk

Ozan Evkaya

School of Mathematics, University of Edinburgh
ozan.evkaya@ed.ac.uk

William P. Kay

School of Biosciences, Cardiff University
kayw@cardiff.ac.uk

Introduction

We live in a data driven world. Much research in STEM departments requires skills in data description, prediction, and inference. These skills are often developed in statistics modules yet increasingly statistical analysis requires computer programming. Therefore, for many undergraduate degree programmes, two closely interrelated but technically distinct disciplines – programming and statistics – must often both be taught. There are different possible approaches to teaching these subjects, most notably whether to deliver them simultaneously or sequentially. In this chapter, we focus on undergraduate level teaching in subject areas that are not primarily programming or statistics. We discuss the advantages and disadvantages to three different approaches and provide guidance on best practice in instructional technique and curriculum design. Case-study examples based on our own teaching experiences are provided throughout to illustrate these strategies.

The topic of teaching statistics to non-specialists has been discussed at length (Kelly 1992; Mustafa 1996; Metz 2008; Gimenez et al. 2013; O'Hara 2016; Bromage et al. 2021). How to do so while acknowledging that programming is an important and related skill has received less focus, which we address here. In all scenarios teachers should be mindful of their own educational context, and, inter alia, what knowledge and skills they want their students to obtain from learning both programming and statistics.

Teaching Statistics Before Programming

This is perhaps the original or "old school" approach. As authors, many of us were taught in this way in our own undergraduate degrees prior to the widespread use of high quality, open-source programming languages (e.g. Python, R). Drawing on our experience, we have found that a key advantage of this approach is its focus on core theoretical concepts in statistics (e.g. probability distributions, uncertainty) without the additional cognitive load associated with technology or implementation. In disciplines where students are likely to need to perform statistical analyses across different contexts and using a range of technologies, this emphasis on foundational statistical reasoning has been particularly valuable.

In practice, we have found that this approach cultivates in students both the ability to conduct 'back-of-the-envelope' analyses in settings where computers are not always available (such as during fieldwork), and enables them to 'think on their feet' when off-the-shelf methods aren't suitable. Importantly, we have observed these behaviours emerge spontaneously rather than through explicit scaffolding.

A second benefit of this approach is that it foregrounds statistics as a discipline in its own right, independent of computation, which can be obscured when programming and statistics are taught together. However, in undergraduate contexts where the primary subject area is not statistics or programming, separating the two can lead to the links between statistics and programming being weakened, potentially reducing insight into the applications of both. This has reinforced for us the benefit of applying statistical understanding once a conceptual foundation has been established. Indeed, if we choose a statistics-first approach then we strive to ensure that students are afforded opportunities to use computational approaches (e.g. programming or

domain-specific interfaces such as STATA) to put their statistical understanding into practice at a later date.

Example

In a first-year geoscience course, statistics was taught independently of programming before moving to a combined approach using Python. Later in the year, during fieldwork, students were asked to measure the size of some ammonite fossils. As they were comfortable with pen-and-paper statistical analysis, students took the initiative to more rigorously characterise the distribution of ammonite sizes, and to estimate the number of observations required to obtain a representative sample. Whilst the statistics course subsequently saw students applying programming to address statistical questions, this in-the-field investigation was facilitated by students first learning to do statistics by hand. Student engagement with 'back-of-the-envelope' statistics in the field emerged autonomously, but as a result we would consider highlighting the potential benefit of this sequencing explicitly for all students.

Teaching Programming Before Statistics

An alternative approach we have adopted in some contexts is to teach programming first, followed by statistics lessons that make explicit use of students' programming skills. Here, students have time to consolidate their programming skills and build their programming mindset. They can then build knowledge and understanding through computational investigation of statistical concepts, before being introduced to formal statistical definitions.

This approach requires more curriculum time than many programmes can accommodate. It also requires that the programming and statistics courses are designed with interaction in mind. For example, using the same programming language in both courses, or intentionally keeping the programming course language-agnostic (e.g. giving examples in multiple languages). We suggest that if this alignment is absent, students may find it more challenging to apply their programming knowledge to statistical contexts.

We have also observed that teaching statistics in a manner which is reliant on the use of programming can present a barrier for students who are less confident in this, particularly early in the curriculum. Indeed, depending on the overall curriculum design, students may have forgotten much of the programming learnt by the time they are learning statistics. We have particularly observed this skill fade at the start of a subsequent academic year following a long summer break. This has led us to conclude that this approach is most effective when programming instruction is closely timed with subsequent statistical teaching, and that particularly between years there is the need to build in time for refresher training. We would avoid a programming-first design when coordination across modules is weak or when a long gap makes knowledge decay likely.

Example

In a second-year course in biomedical informatics, the concept of variance was introduced using a programming-first approach prior to presenting the formal statistical framework of Analysis of Variance (ANOVA). Students were first shown how to use simulations (sampling with replacement) to compare differences between pairs of samples within and across treatment groups. These empirical results were then used to derive the probability of observing a difference between groups and to introduce the concept of variance as a quantitative measure. In the following week, the formal mathematics underlying ANOVA were introduced. Students subsequently learnt how to implement ANOVA using R and, when appropriate, to perform post-hoc pairwise tests to interpret observed differences among treatment groups. We found that this deliberate sequencing helped students to comprehend the underlying theory before learning how to implement the analysis and interpret the output in real-world scenarios. Of course, a trade-off in this approach was the additional contact time needed to run simulations and then later revisit notation, requiring a reduction of coverage elsewhere.

Teaching Programming and Statistics Together

The final approach we have identified is to teach programming and statistics together (Sarvary 2014). In practice, we have seen this implemented in two ways: within a single module, instructors can teach either programming or statistics first and immediately apply it to the other, or the two can be truly interleaved e.g. by introducing new programming constructs and statistical concepts in parallel.

This integrated approach has clear pedagogical advantages. Most importantly, students have reported that they feel able to engage immediately with statistical concepts through computation, and that this can promote a deeper understanding of statistics and its application. Furthermore, this can be supported by introducing new programming approaches when required alongside novel statistical techniques. Students may take a constructivist learning approach to building their statistical knowledge, e.g. by running simulations and adapting code provided in class. For students whose prior negative experience with mathematics presents a barrier to learning statistics (Pletzer et al. 2010), we have found that introducing statistical concepts through code snippets can reduce this anxiety compared to presenting the same idea using formal mathematical notation. That said, our experience also highlights that for some the opposite is true, i.e. that they find it easier to explore concepts programmatically after a formal mathematical introduction.

Despite the benefits of this approach, we have found that the principal challenge of teaching programming and statistics together is the risk of cognitive overload. When both topics are taught concurrently, students are not always able to consolidate new material into their long-term memory and thus the working-memory load is higher (Sweller 2018). In practice, many students will experience hurdles in their learning of both statistics and programming, and these may compound each other to further impede student attainment. We have observed that this effect can be exacerbated for students who are not learning in their native language, as they need to acquire domain-specific vocabulary in two topics simultaneously.

While thoughtful course design - such as carefully and explicitly scaffolding topics, and limiting weekly learning outcomes - can mitigate cognitive overload, it is likely to remain higher when teaching programming and statistics together than with standalone courses (Auker and Barthelmess 2020). Additional trade-offs have also emerged. In some implementations, students' development of broader computational or algorithmic thinking was reduced, since programming was used almost exclusively to perform statistics. For disciplines where programming is primarily used for this purpose, this may be an appropriate outcome. However, where courses aim to set students up to progress to higher-level programming courses, we have found that students benefit from learning to apply programming skills across a broader range of applications to widen their transferable competence. Finally, the choice of programming language and environment can substantially impact learning outcomes, both positively and negatively. This highlights the need for careful course design, e.g. choosing a beginner-friendly programming language when adopting a fully integrated approach. We would therefore recommend avoiding a fully integrated design when class time is too tight, or when support for learners studying in a non-native language is limited.

Example

In a first-year undergraduate course with students from several mathematically oriented degree programmes, concepts such as linear modeling were introduced alongside their computational implementation. During lectures, slides either showed the mathematical and code representations side-by-side or sequentially within the same session, allowing students to make connections across the two representations.

Students then participated in workshops where context-specific exercises with new datasets were given, requiring students to start afresh each time. Students needed to either adapt code snippets based on the given data, or write new code to address each task. We found that regularly changing datasets encouraged transfer of knowledge rather than rote replication. We also learnt that with this style of delivery it was necessary to provide sufficient pauses - either during or between classes - to allow students to fully digest the content from both disciplines, meaning that less time was available for other activities.

Practical Strategies for Teaching Statistics and Programming

Drawing on our teaching experience across a range of programmes, we now summarise a number of practical strategies that we have found effective when teaching programming and statistics, regardless of whether these topics are taught sequentially or simultaneously.

First, we have learnt that embedding the teaching of these topics into the subject matter of the overarching course is critical for student engagement, particularly when programming or statistics are not the principal focus of the degree programme. This embedding may take multiple forms, such as creating appealing visualisations, using authentic datasets, or discussing the conclusions that can be obtained in the students' particular research field. In our experience, this contextualisation helps students understand not only how to perform relevant analyses, but why they are important for their discipline.

Second, statistical methodologies are most effectively taught when also described clearly using a computational approach. For example, while it is helpful to describe the formal mathematical derivation for generating a test statistic, we recommend that this is routinely followed up with the programming approach required to generate the same result. Demonstrating the same concept from both perspectives helps students to consolidate the underlying statistical concepts and to see how statistics are applied in practice, highlighting the close connection between the two approaches.

A third strategy that has emerged from our teaching is to foster a "programming or coding mindset". We found it important to foreground programming as an important transferable skill in its own right, rather than solely being a tool to perform statistical analysis. This involved teaching students how to interact with data programmatically rather than relying on graphical user interfaces or apps alone. In practice, this included providing explicit instruction in code annotation, file paths, naming conventions, and being able to discriminate between characters, numbers, and special symbols. We also found that dedicating time to debugging code – particularly through live programming in lectures – helps to demystify programming and normalise common programming errors.

Finally, our experience strongly suggests that motivating and enthusing students about both statistics and programming is critical. For example, framing computational statistics as a means to increase reproducibility, transparency, and confidence in scientific discoveries helps students appreciate its broader value. We have also found that highlighting the fact that these skills are universally applicable, lucrative, and in high demand in the job market, including beyond academia, is useful for motivating engagement. As an illustrative example, note that the gap in data-driven technical skills reported in 2023 in the UK includes specialist data skills (including statistics), alongside programming and software engineering skills required to manage and analyse datasets (Fearns and Harriss 2023).

Example

In a postgraduate course covering both Python and R, we introduced a recurring activity designed to foster a "coding mindset", called 'Error of the Week'. This feature celebrates errors as learning opportunities rather than sources of embarrassment or frustration. Students were encouraged to share errors on a discussion board, including solutions if applicable. As a course team we responded to every post, and during live sessions the course lead or tutor highlighted a particularly interesting or common error to the class, debugging it live. Over the duration of the course, these discussion boards evolved into a common error glossary for all. We found that these strategies lowered the psychological barrier that programming can often present to students, allowing for more time to be devoted to understanding and interpreting statistical outputs.

Conclusion

We have outlined three approaches for teaching programming and statistics: teaching statistics first; teaching programming first; and teaching both skills together. Our experiences suggest that each approach has clear strengths and limitations, and will be best suited in different settings. In deciding which approach is most suitable it is important to consider the desired learning outcomes, overall programme design, and the cognitive load placed on students, all of which will vary depending on educational context and prior student experience.

Teaching statistics before programming is particularly useful when the priority is to develop strong, transferable statistical reasoning. Teaching programming before statistics can work well when early development of computational confidence is a key goal, and when there is sufficient time and alignment to support knowledge transfer. With careful instructional design, and sufficient curriculum time, many of the affective barriers students experience when learning programming and statistics can be mitigated. Under these conditions, our preferred approach would be to teach programming and statistics together, as this enables students to engage immediately with statistical concepts through computation, and develop a complementary understanding of both. We hope that by highlighting the advantages and disadvantages of different approaches and suggesting practical teaching strategies, this chapter will help educators to make informed decisions about how best to teach and encourage the learning of programming and statistics to their students.

References

Auker, Linda A., and Erika L. Barthelmess. 2020. “Teaching r in the Undergraduate Ecology Classroom: Approaches, Lessons Learned, and Recommendations.” *Ecosphere* 11 (4): e03060. https://doi.org/https://doi.org/10.1002/ecs2.3060.

Bromage, Adrian, Sarah Pierce, Tom Reader, and Lindsey Compton. 2021. “Teaching Statistics to Non-Specialists: Challenges and Strategies for Success.” *Journal of Further and Higher Education* 46 (1): 46–61. https://doi.org/10.1080/0309877X.2021.1879744.

Fearns, Joshua, and Lydia Harriss. 2023. “Data Science Skills in the UK Workforce.” Parliamentary Office of Science; Technology. https://doi.org/10.58248/pn697.

Gimenez, Olivier, Fitsum Abadi, Jean-Yves Barnagaud, Laetitia Blanc, Mathieu Buoro, Sarah Cubaynes, Marine Desprez, et al. 2013. “How Can Quantitative Ecology Be Attractive to Young Scientists? Balancing Computer/Desk Work with Fieldwork.” *Animal Conservation* 16 (2): 134–36. https://doi.org/10.1111/J.1469-1795.2012.00597.X.

Kelly, Martyn. 1992. “Teaching Statistics to Biologists.” *Journal of Biological Education* 26 (3): 200–203. https://doi.org/10.1080/00219266.1992.9655273.

Metz, Anneke M. 2008. “Teaching Statistics in Biology: Using Inquiry-Based Learning to Strengthen Understanding of Statistical Analysis in Biology Laboratory Courses.” *CBE–Life Sciences Education* 7 (3): 317–26. https://doi.org/10.1187/cbe.07-07-0046.

Mustafa, R Yilmaz. 1996. “The Challenge of Teaching Statistics to Non-Specialists.” *Journal of Statistics Education* 4 (1). https://doi.org/10.1080/10691898.1996.11910504.

O’Hara, Robert B. 2016. “On Teaching Ecologists Contemporary Methods in Statistics.” Wiley Online Library. https://doi.org/10.1002/ecy.1325.

Pletzer, Belinda, Guilherme Wood, Korbinian Moeller, Hans-Christoph Nuerk, and Hubert H Kerschbaum. 2010. “Predictors of Performance in a Real-Life Statistics Examination Depend on the Individual Cortisol Profile.” *Biological Psychology* 85 (3): 410–16. https://doi.org/10.1016/j.biopsycho.2010.08.015.

Sarvary, Mark. 2014. “Biostatistics in the Classroom: Teaching Introductory Biology Students How to Use the Statistical Software ‘r’ Effectively.” *Tested Studies for Laboratory Teaching Proceedings of the Association for Biology Laboratory Education* 35 (January): 405–7.

Sweller, John. 2018. “Instructional Design.” In *Encyclopedia of Evolutionary Psychological Science*, 1–5. Springer.