Oecologia Australis (ISSN: 2177-6199)
Ahead of print (https://revistas.ufrj.br/index.php/oa/issue/view/1109/showToc)
Article ID: AO#64848
Published online: 17 FEBRUARY 2026

Challenges of teaching the R programming language to ecologists

THE CHALLENGES AND NUANCES OF TEACHING THE R PROGRAMMING LANGUAGE TO ECOLOGISTS

Maurício Humberto Vancine[1*], *Pavel Dodonov*[2], *Bruno Vilela*[2], *Luisa Diele-Viegas*[3], *Victor Casagrande Souza*[4], *Felipe Alvarez Silva Nunes*[4], *Ana Julia Oliveira Silva*[4], *Beatriz Milz*[5], *Cristina A. Kita*[6], *Marco A. R. Mello*[6] & *Renata L. Muylaert*[7]

[1] Universidade Estadual de Campinas (UNICAMP), Instituto de Biociências, Departamento de Biologia Animal, Rua Monteiro Lobato, 255, Cidade Universitária Zeferino Vaz, Barão Geraldo, 13083-862, Campinas, SP, Brasil

[2] Universidade Federal da Bahia, Instituto de Biologia, Laboratório de Ecologia Espacial, Rua Barão de Jeremoabo, 147, Campus de Ondina, Ondina, 40170-115, Salvador, Bahia, Brasil.

[3] Universidade Federal da Bahia, Instituto de Biologia, Laboratório de (Bio)Diversidade no Antropoceno, Rua Barão de Jeremoabo, 147, Campus de Ondina, Ondina, 40170-115, Salvador, Bahia, Brasil.

[4] Universidade Estadual Paulista (Unesp), Instituto de Biociências, Departamento de Biodiversidade, Avenida 24 A, 1515, Bela Vista, 13506-900, Rio Claro, SP, Brasil.

[5] Universidade de São Paulo, Instituto de Energia e Ambiente, Av. Prof. Luciano Gualberto, 1289, Cidade Universitária, Butantã, 05508-010, São Paulo, SP, Brasil.

[6] Universidade de São Paulo, Instituto de Biociências, Departamento de Ecologia, Rua do Matão, travessa 14, 321, Cidade Universitária, Butantã, 05508-090, São Paulo, SP, Brasil.

[7] The University of Sydney, Faculty of Science, Sydney School of Veterinary Science, Disease Ecology Laboratory, Parramatta Road, s/n, Camperdown, 2006, Sydney, New South Wales, Austrália.

E-mails: mauricio.vancine@gmail.com (*corresponding author); pdodonov@gmail.com; bruno.vilela@ufba.br; luisa.mviegas@gmail.com; vicasagrand@outlook.com; felipe.alvarez@unesp.br; anajuoliveira019@gmail.com; milz.bea@gmail.com; c.akemikita@gmail.com; marmello@usp.br; renata.muylaert@sydney.edu.au

Abstract. The R programming language, esteemed for its statistical prowess, data management, data visualization, and machine learning capabilities, has become a cornerstone of data science applied to Ecology. Its open-source nature and broad array of analytical tools have garnered a user base, mainly among ecologists. Nevertheless, instructing R to biology and ecology students

presents some key challenges. Here, we delve into the nuances of teaching R in graduate and undergraduate ecology courses, focusing on data wrangling, data visualization, and statistics. We provide insights from educators at various career stages, with a strong focus on the Brazilian context. No single way of teaching R fits all situations; however, some general guidelines can be provided and followed. Before starting this educational journey, the foundations must be addressed, including infrastructure, hardware, and software. Once these prerequisites are secured, students confront the intricacies of R's programming landscape. Abstract concepts, coding idiosyncrasies, package compatibility, and the interplay between code and data need to be mastered. The narrative progresses to the challenge of interpreting R's outputs and integrating them seamlessly into statistical and ecological analyses. Additionally, we consider the impact of structural challenges and global pressures on R education, such as the COVID-19 pandemic and the evolving influence of artificial intelligence tools. In conclusion, exploring R within ecology envisions a future where professionals possess the analytical skills to unlock innovative solutions and applications. As this educational journey concludes, we present our perspective on the future of R teaching and how it can help develop our science.

Keywords: Brazil, coding, ecology, data science, higher education, programming, science education.

Resumo. A linguagem de programação R, estimada por sua habilidade estatística, gerenciamento de dados, visualização de dados e capacidades de aprendizado de máquina, tornou-se uma pedra angular da ciência de dados aplicada à Ecologia. Sua natureza de código aberto e ampla gama de ferramentas analíticas conquistaram uma base de usuários, principalmente entre os ecólogos. No entanto, instruir R para estudantes de biologia e ecologia apresenta alguns desafios-chave. Aqui, mergulhamos nas nuances de ensinar R em cursos de graduação e pós-graduação em ecologia, focando na manipulação de dados, visualização de dados e estatísticas. Fornecemos percepções de educadores em várias fases da carreira, com forte foco no contexto brasileiro. Não existe uma única maneira de ensinar R que se adapte a todas as situações; no entanto, algumas diretrizes

Oecologia Australis (ISSN: 2177-6199)
Ahead of print (https://revistas.ufrj.br/index.php/oa/issue/view/1109/showToc)
Article ID: AO#64848
Published online: 17 FEBRUARY 2026

gerais podem ser fornecidas e seguidas. Antes de iniciar essa jornada educacional, é necessário abordar os fundamentos, incluindo infraestrutura, hardware e software. Uma vez garantidos esses pré-requisitos, os estudantes enfrentam as complexidades do ambiente de programação do R. Conceitos abstratos, idiossincrasias de codificação, compatibilidade de pacotes e a interação entre código e dados precisam ser dominados. A narrativa progride para o desafio de interpretar os resultados do R e integrá-los perfeitamente em análises estatísticas e ecológicas. Além disso, consideramos o impacto dos desafios estruturais e as pressões globais na educação em R, como a pandemia de COVID-19 e envolvendo a influência das ferramentas de inteligência artificial. Em conclusão, explorar o R na ecologia vislumbra um futuro em que os profissionais possuem as habilidades analíticas para desbloquear soluções e aplicações inovadoras. À medida que essa jornada educacional termina, apresentamos nossa perspectiva sobre o futuro do ensino de R e como ele pode ajudar a desenvolver nossa ciência.

Palavras-chave: Brasil, codificação, ecologia, ciência de dados, ensino superior, programação, educação científica.

INTRODUCTION

Discussions about a canonical curriculum in Ecology (Dormann & Mello 2023) have sparked debates on how to teach natural phenomena, problem-solving, and programming, while also highlighting the need to update the data analysis methods taught in undergraduate and graduate courses. Use of the R programming language (R Core Team 2024) for ecological data analysis has been rising continuously in the past two decades (Touchon & McCoy 2016) and has become an important skill for ecologists (Auker & Barthelmess 2020). Teaching R, however, may be challenging, especially for students with little or no programming background (Medeiros *et al.* 2019, Auker & Barthelmess 2020).

Traditionally, ecological data analysis in Brazil mainly relied on proprietary software, in most cases costly and inflexible. The turn towards programming began to change this scenario in the early 2000s, when R 1.0.0 was released to the public (https://cran.r-

project.org/bin/windows/base/old). Brazilian institutions slowly began integrating R into their curricula, although not systematically or uniformly. To facilitate the change, workshops, courses, and online resources proliferated, making R learning tools accessible to students and professionals worldwide. Today, R is a staple of ecological research (Foulkes 2009, Lai *et al.* 2019, Mello & Muylaert 2020, Da Silva et al. 2022, Wickham *et al.* 2023), with its use growing also in the classroom. Using R can empower new generations of ecologists to conduct creative, robust, and reproducible research, fostering a more open, collaborative, and innovative scientific community.

Despite this game-changing potential, teaching R to ecology students presents many challenges, largely shaped by student preconceptions about mathematics, statistics, and programming (Auker & Barthelmess 2020). Many students enter ecology with a strong passion for natural history, but also a considerable degree of anxiety related to quantitative skills. This can create a mental hindrance, leading students to believe that proficiency in R and programming is beyond their reach.

Another difficulty in teaching R can be described as the "point-and-click mindset" versus the "command line mindset". The use of friendly graphical user interfaces (GUIs) fosters clicking through menu items rather than understanding how the underlying code is built. While GUIs make technology widely accessible, they limit deeper understanding (Selwyn 2022). This mindset seems to be even more prevalent among younger generations due to their constant exposure to GUIs on smartphones and tablets. Reliance on GUIs is reinforced by educational tools, software trends, and a cultural emphasis on instant gratification (Yin & Shen 2024), and can lead to low engagement in programming courses. Students may also find programming intimidating and prefer sticking to more user-friendly tools (Farrell & Carey 2018). To address this issue, educational programs are incorporating both graphical and command-line interfaces to improve computational literacy, fomenting the transition to a "command line mindset", where students write and understand code.

Ultimately, in our understanding, the key to teaching R to ecology students lies in addressing their preconceptions and insecurities, providing them with a solid computer science

foundation and effective troubleshooting strategies. By doing so, instructors can empower students to overcome their initial apprehension and develop confidence in programming. This not only enhances their capacity to conduct sophisticated ecological research but also broadens their skill set, making them more versatile scientists with higher employability. Through structured guidance and supportive teaching practices, students can transform their initial trepidation into a strong proficiency in R, leveraging its powerful capabilities to advance their careers. The successful teaching of R to Ecology students hinges on a well-rounded approach that combines solid foundations, advanced analytical skills, and active learning strategies. In the next sections, we discuss some ways to achieve these goals. We intend our suggestions to be applicable to a wide range of course levels, from introductory undergraduate courses to students without previous contact with R or statistics, to more advanced Masters and PhD courses.

Increasing computational literacy

To help students overcome these problems, instructors may emphasize the practical utility of R in ecological research, demonstrating how it simplifies complex data analysis and visualization, ultimately fomenting their analytical independence and making their scientific inquiries more robust and insightful. This may be especially useful in introductory courses or for students who didn't have previous contact with R or programming. Instructors could also show students that programming literacy is critical to understanding the ecological literature, as most analyses, including methods, are currently coded in R (Zuur et al. 2010, Zuur & Ieno 2016). To succeed as professional ecologists, it is imperative for students to understand how data are analyzed in key scientific papers.

A foundational understanding of computer science is also essential for learning R effectively. This includes familiarity with concepts such as operating systems, file types, directory structures, data types, control structures (e.g., loops and conditionals), language vocabulary and syntax, functions and arguments, and the logic of how computers execute code. Specifically for R, this also includes understanding functions and packages. Functions are pieces of code that

perform a specific task, with a structure like *function_name(argument1, argument2, ...)*. We can draw parallels to the concept of a mathematical function *f(x)*: "f" is the function that performs an operation to modify x; "x" represents the input data or arguments. Packages are a collection of R functions, sample data, and documentation that describes how to use them (Wickham & Bryan 2023). They contain functions designed to perform specific tasks beyond those installed in Base R. There are thousands of packages (discover for yourself in R: `*nrow(available.packages())*` or at this link: https://cran.r-project.org/web/packages) for a wide range of purposes; the most popular R packages among ecologists include *lme4*, *vegan*, *nmle*, *ape*, and *MuMIn* (Lai *et al.* 2023). Packages can be available as revised versions on the *Comprehensive R Archive Network* (CRAN), or as development versions on GitHub and other versioning platforms.

Given that most beginner ecology students (e.g., undergraduates taking their first ecology courses), as well as some more advanced ones (e.g., grad students who had not had previous contact with R), need to gain such a background (Braga *et al.* 2023), instructors should start with these fundamentals before delving into more complex R particularities. Introducing these concepts through relatable ecological examples can bridge the gap between theoretical knowledge and practical application, fostering a clearer understanding of programming logic.

Programming also involves dealing with bugs and errors, meaning that students must develop a systematic approach to debugging and troubleshooting. Encouraging a "command line mindset" that views errors not as failures but as learning opportunities can significantly enhance their problem-solving skills (Kruger & Dunning 1999). Instructors should teach students how to interpret error messages, use debugging tools, and employ strategies such as code commenting and incremental testing to identify and fix issues (Zuur *et al.* 2010). Collaborative learning environments where students can share and discuss shared challenges provide additional support and reduce debugging frustration (McCartney 2016).

Types of courses

Oecologia Australis (ISSN: 2177-6199)
Ahead of print (https://revistas.ufrj.br/index.php/oa/issue/view/1109/showToc)
Article ID: AO#64848
Published online: 17 FEBRUARY 2026

Before teaching R, it is important to define the type of course being taught: is the course focused on R itself (e.g., an R programming course), or does it use R as a tool to teach other subjects (e.g., a statistics course that uses R, but that could also use "point-and-click" software)? We can also have a middle ground – a course aimed at teaching the concepts and their applications in the R environment, in which learning R is an essential part of the course, but the course itself is not limited to it. Thus, it is essential to clarify how R fits into the course—is it the subject or a tool?—and to communicate this to the students early on. It is also essential to have clear objectives in mind and an assessment of the infrastructure available, both at the teaching institution and at students' homes. For instance, not all universities have computer labs that students can use at will, which enormously facilitate learning R, especially for students who come from low-income families.

Teaching the R programming language to ecology students also requires a structured approach that balances basic and advanced courses while incorporating active methodologies to enhance learning outcomes (Deslauriers *et al.* 2019, Hoffman & Wright 2024). To succeed in learning R, students must be prepared to engage deeply with both the conceptual and practical aspects of the language. Basic courses focused on R should lay a strong foundation by introducing students to fundamental programming concepts and R syntax (see, for instance, "Use of R Language for Data Analysis": https://uspdigital.usp.br/janus/Disciplina?sgldis=BIE5782& and "R Programming": https://www.coursera.org/learn/r-programming). These courses typically cover data types, the logic of functions, their parameters, and arguments, data manipulation, and basic R visualizations. Abstract concepts should be tied to ecological data and research questions, for example, small empirical datasets can be used for creating objects and applying functions to them (an example used by one of us consists of ten values of bivalve cover associated to ten values representing water quality, collected in the same city where the university is located, at sites that are known to most of the students). Other data already organized as Data Papers (e.g., ATLANTIC SERIES - https://esajournals.onlinelibrary.wiley.com/doi/toc/10.1002/(ISSN)1939-9170.AtlanticPapers), can also be used, mainly because they are usually extensive (there are many

rows and columns), may need to join different tables, and almost always have missing data and even typing errors that can be interesting data manipulation challenges for students. Hands-on practice, with regular assignments involving real-world datasets, is also crucial.

Students in more advanced R courses (see, for instance, "Regression Modeling in Practice": https://www.coursera.org/learn/regression-modeling-practice), should encounter more complex topics such as advanced statistical methods, modeling, and the writing of user-defined functions. These courses demand a higher level of engagement and problem-solving skills. To succeed, students must be proficient in the basics and ready to tackle complex analyses. Advanced courses should emphasize the application of R to exciting ecological problems, encouraging students to develop customized scripts and functions. The ability to independently troubleshoot and optimize code is critical at this stage, as is the capacity to interpret and communicate complex results effectively. More advanced courses can include writing packages, which require advanced knowledge of functions and their descriptions.

Conversely, a course that uses R as a tool and not as the subject may emphasize a small set of basic skills and functions, enabling students to take a first step in learning R. These skills may include reading spreadsheets in R, basic descriptive statistics, data visualization, and simple statistical tests, focusing on real-world data and applications. Thus, even without gaining detailed knowledge of R syntax and data structures, the student may be able to use R for basic analysis. Still, some understanding of the underlying language (e.g., what are objects and functions) is needed, and, if possible, teaching the creation of functions from scratch can be highly beneficial.

In addition to basic and advanced R courses, this programming language can also help teach truly ecological courses. For instance, at the University of São Paulo, Brazil, there are labs about animal ecology written as R notebooks in an in-person undergraduate course (see "Advanced Topics in Animal Ecology": https://uspdigital.usp.br/jupiterweb/obterDisciplina?nomdis=&sgldis=bie0315). In these labs, students reproduce data analyses found in ecological papers and are expected to link concepts and analyses through a combination of lectures, readings, quizzes, and labs. The same strategy is used

by us in a remote course on network science, which also includes R notebooks in its labs (see "Ecological Networks": https://www.coursera.org/learn/redesecologicas).

Active methodologies play a pivotal role in teaching all those courses. These approaches, which include flipped classroom, peer learning, and project-based learning, require students to be protagonists in the learning process (Deslauriers et al. 2019, and see proposal in Cap. 1 from Da Silva et al. 2022 - https://analises-ecologicas.com/cap1#como-ensinar-e-aprender-com-esse-livro). Students must be willing to engage proactively, work collaboratively, and embrace the iterative nature of coding. We realize that not all students have this level of motivation; however, in our experience, classes with hands-on experience, in which the students are able to solve problems and see their code working, can boost engagement. Furthermore, keeping students motivated to learn programming can be a significant challenge, as not everyone has a natural aptitude or aptitude for logical and abstract concepts. Therefore, we believe that practical examples and applications are good motivators for students to maintain a high level of interest.

In a flipped classroom, students might watch recorded lectures, read papers on their own, and then engage in hands-on coding exercises during in-person class time. In project-based learning, students may organize themselves into groups and work on developing a research project or solving a problem on their own, with "scaffolding" provided by the instructors. The group projects may be well-defined or more open, simulating real-world ecological research scenarios. They may also be solved in a single lesson or span several weeks. By integrating these methodologies, educators can create a dynamic and supportive learning environment that encourages students to take ownership of their learning journey.

Finally, there may also be huge variation in learning speed and skill level among the students, and strategies for maximizing the learning experience for all students, such as standardizing classes (known in Portuguese as *aulas de nivelamento*), may increase the odds of providing a uniform experience for everyone. Working on basic and advanced activities encompassing different skill levels in tandem is also possible. In addition, more skilled students,

or those with previous experience, can be asked to be "one-time teaching instructors" and help their colleagues.

Teaching computational thinking

One of the most significant challenges students face when learning R is not mastering the language's syntax (we all know that even advanced users resort to search engines (e.g., Google) or generative artificial intelligence (e.g., ChatGPT) regularly) but understanding how to approach complex problems (Martinez-Blasco *et al.* 2023). Students often freeze at the initial stages because they attempt to address the entire problem simultaneously and do not know where to begin. The key issue is operationalizing the issue into smaller difficulties that are easier to solve, a process also called "computational thinking" (Wing 2006). We suggest a simple, practical structure adapted to help students break R problems into smaller, more manageable parts (see Wu *et al.* 2024 for general propositions): (1) define the problem; (2) define the input and output; (3) break down the processing steps; and (4) translate the steps into code.

Define the problem: The first step is to define the problem itself and operationalize it. This can be straightforward, such as creating a function to calculate species richness in multiple locations, or more abstract, such as describing the biodiversity of several protected areas. In the latter, we need to clarify the biodiversity metrics to be utilized before attempting to address the question itself. Therefore, we must operationalize our variables (clearly define and specify the proxies for abstract concepts), just as we do in hypothesis testing. This is a skill that can be explored from beginner to advanced courses, with problems of different levels.

Define the input and output: The next step is to determine the input data, or information, that should be provided by the user to solve the problem. For example, if we want to calculate the species richness of two or more locations, we need a community matrix, where rows represent locations and columns represent species, and each value denotes the abundance or presence of a species at a given location. We then define the output, which should be defined now, as one should be able to clearly foresee the structure of their result before obtaining them. This enables

individuals to align their efforts with their ultimate objectives, fostering efficiency and effectiveness in problem-solving. In our example, the expected output is species richness per location. Defining the input and output beforehand also allows any code to be modular, composed of independent, reusable units or modules, each with a specific function, which increases efficiency and collaboration in science (e.g., Kass 2018).

Break down the processing steps: We can now move forward to breaking down the problem into processing steps. Instead of asking students to calculate species richness, we need to guide them on seeing the larger task as a sequence of smaller, manageable steps. One approach is to provide them with a simplified version of their input data and ask them to calculate species richness by hand, paying attention to the mental steps used in the calculation and writing them down, including input and output information. This may result in something like:

1. Input data: Community matrix.
2. Identify the species with abundance values greater than 0 at a given location.
3. Count the number of unique species identified in step 1.
4. Write/store this number.
5. Repeat this process for all locations.
6. Output data: Species richness values per location.

These written steps to perform a task are also known as a pseudocode: a simplified, informal protocol written in natural language, which is used to outline the logic and steps of an algorithm in a more friendly way (Petre & Winder 1988). It is not tied to any specific programming language and avoids complex syntax, useful for planning and communicating how a script or function will work before writing the actual code (Bellamy 1994).

Translate the steps into code: Now that the problem of creating a function to calculate species richness across multiple locations has been largely decomposed into manageable steps, its pseudocode can be translated into actual R code. The pseudocode can also serve as

documentation, enhancing the code's clarity and reproducibility. Students can then test the function, optimize its performance and even explore entirely different algorithms. For instance, students could improve the algorithm by taking vectorization and avoiding step 5, which requires a more complex structure involving a loop (a control structure for a set of instructions to be executed repeatedly over a sequence of values). This structured approach also equips students with essential problem-solving skills, such as abstract thinking, systematic planning, and iterative refinement, which are invaluable for tackling a wide range of challenges they may encounter in their careers.

Online teaching

The social isolation caused by lockdowns during the COVID-19 pandemic has greatly impacted higher education, forcing a switch from in-person to online and then hybrid teaching in universities worldwide (Chow *et al.* 2021). After the pandemic, many teaching activities continued remotely or semi-remotely, often using the experience gained during the pandemic. Instructors who had limited experience with digital teaching resources (SEAD-UFBA 2020b) had to convert in-person courses to full remote versions overnight. Below, we provide some recommendations for such courses based on our experience with online teaching before, during, and after COVID-19.

The first step is to assess which tools are available for which students. Not all students have access to computers at home. For example, a survey performed by the Federal University of Bahia in 2020 (SEAD-UFBA 2020a) showed that approximately a quarter of its students had little to no conditions for online learning, only 18% had access to a desktop personal computer and 72% to a notebook, and approximately 1% did not have access to any digital device. Internet access was limited for approximately 20% of the students. These data provide a glimpse of the dire situation faced by many students, especially in developing countries. Although the conditions vary widely among universities, educators must understand that not all students have access to a computer with fast, stable internet, and those with access might need to share the device with other people

Oecologia Australis (ISSN: 2177-6199)
Ahead of print (https://revistas.ufrj.br/index.php/oa/issue/view/1109/showToc)
Article ID: AO#64848
Published online: 17 FEBRUARY 2026

in their household or neighborhood. Thus, for undergraduate and graduate courses in vulnerable communities (see **Structural challenges)**, we recommend first performing an institutional survey with the students to assess what infrastructure they have available before choosing learning strategies and tools. For outreach courses, an alternative would be to list the equipment required. It is also important to keep in mind that some packages may be too memory-consuming for older computers, and the minimal requirements for running the course should be mentioned in the syllabus, including online alternatives such as Posit Cloud (formerly RStudio Cloud) (https://posit.cloud).

In addition, most students will not have two screens available. If they must follow the instructor's demonstration and write their own code at the same time, they are likely to become lost along the way. One alternative is to separate teaching and coding/writing moments: the instructor first makes a demonstration, and then the students are given time to work on their own code, based on their notes. Providing preparation material in advance as PDF, R and R markdown/quarto files with task checklists is highly recommended.

Another barrier is that whereas in an in-person course the instructor can move around the classroom by checking the code, giving suggestions and correcting errors, this is challenging in an online class, especially if the number of instructors and assistants is not enough to reach all students. Commonly used online tools tend to lack interactivity and engagement, making it difficult to get help or communicate issues due to limited attention (often the student is not allowed to share their screen by default). Internet connection issues disrupt learning and frustrate students and instructors. One way to deal with those challenges is to use smaller break rooms where students can enter, share their screen with an instructor or assistant, and get help to solve their issues. Another alternative is for students with difficulties to share their screen with the entire class, so everyone, including the instructor, can help solve their issues. This latter alternative can be particularly interesting, as by helping others fix their colleagues' issues, students can improve their own skills.

Oecologia Australis (ISSN: 2177-6199)
Ahead of print (https://revistas.ufrj.br/index.php/oa/issue/view/1109/showToc)
Article ID: AO#64848
Published online: 17 FEBRUARY 2026

Group activities enable the participation of students having access only to smartphones or tablets, as we observed in our courses during the pandemic. Instead of each student writing their own code, they can be divided into smaller groups, in which one student shares their screen with the group and everyone contributes. The instructor may interact with each group for a short time to guide the students. This last approach is especially interesting for active strategies, such as project-based learning. After a brief explanation, the students may be given a problem to solve in R, such as analyzing a given dataset to answer a question. The students then divide themselves into groups, each group using a different online room or a separate call. While each group works on the problem, the instructor interacts with each group in question, and at the end of the lesson, all groups meet and share their solutions.

Finally, it is important to have a well-organized list of online resources, such as texts and video-lectures, for students to be able to keep learning beyond class time. This is valid also for in-person teaching, but even more so for synchronous online courses (see **Structural problems)**. Fully asynchronous courses are also an option, especially for outreach. For these courses, we recommend using a combination of videos, texts, quizzes, and labs for the students to solve, individually or in groups.

Tutoring

Since many students have limited or no programming experience, having guest lecturers and teaching assistants to aid the main instructor can greatly improve the learning process. First, students might feel more comfortable asking questions to a tutor than to a professor, because tutors may offer personalized attention and a more approachable environment, especially in larger, more formal class settings led by professors. Even in smaller classes, with 15–20 students, it may be impossible for a single instructor to give attention to all students. In these cases, tutors can offer office hours to individual students or groups and stimulate peer-learning sessions, enhancing the students' understanding while also fostering a sense of community and teamwork. In addition, beginners may struggle with basic R syntax, such as missing parentheses, quotation

marks, or incorrect function names. R error messages can also be cryptic at first, making debugging challenging, time-consuming, and frustrating. By quickly identifying syntax problems, tutors can optimize learning.

For students with little knowledge of statistics and experimental/sampling design, understanding and applying statistical concepts to data analysis is challenging. However, the hands-on experience with R can also aid in learning these concepts, especially if this experience is guided by tutors. Tutors can assist students by offering help sessions on concepts and applications through data analysis and help students understand what has been asked in their assignments. If needed, tutors can also encourage good data management practices (Broman & Woo 2018) during these sessions, such as the principles of tidy data (Wickham 2014), to make their sheets organized and human- and machine-readable (Harrison 2014, Cooper & Hsing 2017). Tutors can also help with the biological interpretation of statistical model results.

Although there are many resources available to learn R on the internet, the quality, and accessibility of these resources varies largely. Tutors can guide students by indicating good sources of information, including books, online forums, and online courses (see Supplementary Material). The exact nature of this contribution may be something like providing a pseudocode for the students to apply; aiding the students' reasoning by asking guiding questions; help with resolving R-specific issues; or something else, depending on the course's nature.

Finally, on a related topic, peer-learning outside the classroom is also great. An example is the GERE (*Grupo de Estudos em R e Estatísticas*) [https://github.com/grupo-estudos-r-estatistica] from Unesp – Rio Claro, a study group in statistics and R that was established in 2024 in response to strong resistance to mathematical thinking in biological undergraduate courses. Initially supported by a few members, the group's attendance grew significantly through social media and word of mouth, reaching an average of twenty participants per meeting. Guided by the book "Ecological Analysis in R" (Da Silva *et al.* 2022), the group conducted hands-on sessions twice a week, at two times of the day, morning and evening, with the same content, covering topics from basic R to linear model analysis, and encouraged sharing of research projects.

Oecologia Australis (ISSN: 2177-6199)
Ahead of print (https://revistas.ufrj.br/index.php/oa/issue/view/1109/showToc)
Article ID: AO#64848
Published online: 17 FEBRUARY 2026

Structural challenges

Teaching R programming to ecology students in Brazil and other geopolitically peripheral countries may be hindered by lengthy structural challenges stemming from a range of infrastructural issues, political and economic instability, and logistical constraints. For instance, universities in Brazil commonly struggle with intermittent internet connection, which disrupts programming classes, especially if package updates and data download are required. This also limits synchronous online courses, in which, due to internet instability, power shortages, and other unpredictable issues, students may be unable to attend some lessons.

Another problem is outdated or limited computer labs (including limited number of chairs and desks) and insufficient technical support. Many computer labs are equipped with outdated machines that cannot efficiently run modern software, including the latest versions of R and its multitude of packages. This forces students and instructors to spend valuable class time troubleshooting technical issues rather than focusing on programming concepts and applications.

Lack of adequate IT support (technical help with computer systems) is also a challenge, as dedicated personnel may be understaffed or unavailable, meaning that technical issues cannot be resolved promptly, leading to delays and interruptions during classes. Moreover, instructors often do not have administrative privileges to update or install software on university computers, limiting their possibilities of troubleshooting. This is particularly problematic when software updates or new package installations are required to run certain R scripts. Setting up programming environments, troubleshooting software issues, and ensuring that all students can access the necessary tools thus become formidable tasks.

The political instability in Brazil (see https://dynamicecology.wordpress.com/2017/11/28/guest-post-doing-ecology-on-a-rollercoaster-in-brazil for a perspective on the topic) frequently leads to strikes and protests by both faculty and students – often at the end of academic semesters, when advanced topics in R programming are typically covered. Because of this, crucial portions of the curriculum may be

rushed or superficially covered. Faculty and student strikes can also delay grading and feedback. Without timely feedback, students may struggle to keep up with course requirements and fall behind in their understanding of complex programming concepts. Sometimes, entire chunks of the curriculum are not delivered at all due to time constraints.

Finally, logistical issues, including transportation and safety, impact student attendance and participation, especially for nocturnal courses. Many students face challenges commuting to campus due to unreliable public transportation and safety concerns (in extreme cases, safety concerns lead to classes being suspended - https://www.correio24horas.com.br/minha-bahia/cursos-na-ufba-suspendem-aulas-em-meio-a-violencia-no-alto-das-pombas-e-calabar-0923). These issues result in irregular attendance, causing students to miss critical classes and disrupting the continuity of their learning. The lack of continuity in attendance means that students often miss important lectures and hands-on sessions. In a subject like R programming, where concepts build on each other, missing even a few classes can significantly hinder a student's performance.

Addressing these challenges requires consistent efforts to improve and maintain infrastructure, provide reliable IT support, and mitigate logistical and safety concerns in campus and public transportation, as well as great planning skills and adaptability on the instructors' part.

Diversity and inclusion

We view inclusion and equity as a continuous process involving structures, systems, guidelines, and our teaching practice. Applying the equity and inclusion principles in our teaching practice can vary according to institutional structure and systems, and guidelines for the tertiary sector may vary in terms of curriculum delivery and assessments (Table S4). Acknowledging the multiculturalism and extensive diversity of students in Brazil is a step forward in inclusion, and advancing efforts to disseminate equitable practices should be at the forefront of developing guidelines for teaching practice in Brazil. Barriers to computational literacy are grounded in overall literacy inequality in Brazil, and they extend to language barriers, as much material for R

programming and most function descriptions are in English. Moreover, gender disparities are common in undergraduate programming (Forrester *et al.* 2022).

A valuable initiative tackling such issues is R-ladies (https://rladies.org), a global organization aiming to promote gender diversity in the R community. The organization was founded in 2012 by Gabriela de Queiroz—a Brazilian woman studying in the US at that time. The organization has since grown and there are chapters (local groups) in all continents, including active chapters in Brazil, such as R-Ladies São Paulo (https://rladies-sp.org), R-Ladies Goiânia (https://www.rladiesgyn.com), and R-Ladies Belo Horizonte (https://rladiesbh.com.br). The chapters organize free activities, including online or in-person events, study groups, and book clubs, focused on teaching R. The chapters also provide free online resources including comprehensive tutorials, coding exercises, and recorded workshops on a wide range of topics, from basic R programming to spatial data analysis. Additionally, R-Ladies maintain active blogs and social media channels where members promote inclusion and diversity and share their knowledge, experiences, and updates on the latest developments in the R community.

Additional considerations and resources

IDEs e text editors

The R programming language comes installed with a command line console, called RGui on Windows, and directly on the terminal on Unix operational systems (GNU/Linux and macOS). However, there are other ways to write R code through Integrated Development Environments (IDEs), which assist with code completion, automatic code generation, error detection, and syntax highlighting. Some IDEs have been specifically developed for R, such as RStudio (RStudio Team 2020), which combines an R console, a syntax-highlighting editor with an autocomplete function, and tabs for plots, history, git, and help. RStudio may be taught alongside R, but it is important to distinguish between the two. Recently, Posit is working on a next-generation data science IDE called Positron™ (https://github.com/posit-dev/positron), which shifted to a language-agnostic IDE built by Posit (formerly RStudio). Other IDEs are shown in Table S1.

Oecologia Australis (ISSN: 2177-6199)
Ahead of print (https://revistas.ufrj.br/index.php/oa/issue/view/1109/showToc)
Article ID: AO#64848
Published online: 17 FEBRUARY 2026

tidyverse and tidymodels

R has undergone adaptations in recent years, including a new approach to data manipulation, called Tidy Data (Wickham 2014), which focuses on data cleaning and organization. Building on these ideas, the tidyverse (https://www.tidyverse.org) was implemented in R as a collection of packages (e.g., *dplyr, ggplot2, magrittr, readr*, *tibble*, and *tidyr*) designed for data importation, manipulation, exploration, visualization, analysis, and communication (Wickham *et al.* 2019). This set of packages significantly alters data flow operations in R, introducing the concept of piping (*%>%*) and functional programming (e.g., *purrr*, Wickham 2019). Following the same approach, *tidymodels* (Kuhn & Wickham 2020) (https://www.tidymodels.org) is a collection of packages for modeling and machine learning.

There are questions on whether these modifications are here to stay or if it might be beneficial to revert some aspects to the Base R version (Carscadden & Martin 2022, Çetinkaya-Rundel et al. 2022). This poses a significant question: should R be taught starting with Base R, focusing on its functions and syntax, or is it more advantageous to begin with the *tidyverse*? We believe there is no definitive answer to this question yet, but we suggest, if time and equipment permit, starting with Base R and then proceeding with tidyverse (Da Silva *et al.* 2022). Many concepts used in the *tidyverse* can be confusing if the fundamental classes and object types are not explained in Base R first.

ggplot2

Within the *tidyverse* approach, *ggplot2* (Wickham 2016) is a widely used package for creating graphics (Nordmann *et al.* 2022). Based on the grammar of graphics (Wilkinson *et al.* 2005), *ggplot2* package has its own syntax based on layers, requiring specific functions which are connected using the + operator. The dichotomy between Base R and the tidyverse also arises here, with teaching challenges related to the plotting functions in Base R and the syntax and functions of *ggplot2*. *ggplot2* codes can quickly become lengthy and sometimes difficult for beginners to

515 comprehend due to *ggplot2*'s layered approach and the need for explicit specification of plot
516 components like themes, data, aesthetics, and geometries. As before, we believe that there is no
517 definite answer to whether visualization should be taught using Base R or *ggplot2*, but we
518 recommend exploring both if time allows.
519
520 *Reproducibility, clean code, and style guides*
521 Reproducibility can be viewed as a gradient (Figueiredo *et al.* 2022) and includes
522 considerations of required packages, data, code, and a description of the R environment. Code
523 readability is key for guiding the reader, but if the code does not run, its usefulness might be way
524 more limited. Some R programmers like to minimize code as much as possible, which can
525 compromise understanding. Conversely, long code may be viewed as more prone to error *due to*
526 *the simple fact that* writing more characters means more chances for typos. The sweet spot for
527 high understanding of code at a particular analysis depends on style, but it will strongly rely on
528 the code working, code commenting, and adequate metadata.
529 Clean code practices are essential for readability and maintainability (Martin 2011), and
530 style guides for Base R (https://google.github.io/styleguide/Rguide.html), and the *tidyverse*
531 (https://style.tidyverse.org) can help standardize codes. However, teaching the creation of
532 functions from the beginning of courses can be highly beneficial for students to advance their
533 understanding of the language's structure and syntax.
534
535 *R Markdown and Quarto*
536 R Markdown (https://rmarkdown.rstudio.com) is a dynamic document format that allows
537 the integration of text, code chunks, and code output, all within a single document. It also enables
538 the creation of interactive documents, sites, and applications with Shiny, which can be responsive
539 to new data (Xie *et al.* 2019). Quarto (https://quarto.org) is an open-source scientific and technical
540 publishing system, and it is the new generation of R Markdown. Both R Markdown and Quarto
541 can enhance the reproducibility, transparency, and dissemination of research results. They can be

used both for teaching and preparing teaching material, and as a way for students to integrate text and code to build their own study material.

Version control (git and GitHub)

Version control is a system that tracks changes to files and directories over time, enabling multiple users to collaborate on projects (Braga *et al.* 2023). Git (https://www.git-scm.com) is the most common software for managing version control and supports integration with remote repositories, such as GitHub (https://github.com); both are also integrated into RStudio. It tracks changes, allowing for rollback, comparison, conflict resolution, and collaboration through branches and mergers. Version control provides backup through commits and remote repositories, facilitating peer review via pull requests. Teaching version control concepts is crucial to prepare students for collaborative projects.

Generative Artificial Intelligence (AI) tools

The rise of large language model-based chatbots (generative AI) from late 2022 on (e.g., https://openai.com/index/chatgpt) marked a transformational milestone in computational tools and natural language understanding, massively influencing various domains, including education (Cooper *et al.* 2024). Generative AI models quickly create and review content (computer code, narrative text, image, film, and audio), including data insights and complex code workflows, from iterative prompts provided by the user, based on the patterns the tool has learned from and the data it was trained on. Using generative tools in teaching R for students can enhance the learning experience by providing personalized, real-time assistance and explanations (Yilmaz & Karaoglan Yilmaz 2023). These tools can help students grasp complex concepts and syntax in R by offering immediate feedback, logical explanations, cross-language examples (such as translating pseudocode to R and then to Python) and troubleshooting advice. They can also build workflows to tackle problems relevant to ecological studies. All this can foster a deeper

understanding of R, streamline the learning process, and help empower students to apply their coding skills to real-world ecological research.

Nevertheless, using generative AI for teaching can lead to problems such as over-reliance on AI (compromising critical thinking and independent problem-solving skills), concerns about academic integrity through potential plagiarism, reduced engagement with foundational concepts, decreased motivation for learning and retaining knowledge, and incorrect information, known as hallucinations, provided by AI (see, for instance: https://revistapesquisa.fapesp.br/universidades-brasileiras-discutem-regras-de-uso-de-inteligencia-artificial/). Unequal access to AI tools can further exacerbate disparities in learning outcomes, while ethical questions persist regarding content ownership and fairness in academic settings. Balancing the benefits of AI in education requires addressing these challenges and generating clear institutional guidelines to ensure responsible and effective use in the classroom.

Reactions to AI by university faculty vary from altogether banishing it to wholeheartedly embracing it and making its use mandatory in assignments (see https://revistapesquisa.fapesp.br/pesquisadores-usam-inteligencia-artificial-em-tarefas-academicas/). The discussions include academic integrity, ethical conduct, what the learning experience is about; and what is expected from students in their assignments and learning outcomes in face of a mutable job market. For students, this involves submitting their own original work in assignments, properly acknowledging sources when using existing knowledge, adhering to ethical guidelines in research, and following instructions. Breaches of academic integrity undermine the trust in academic standards and the value of education. Still, upholding academic integrity is challenging due to pressures to perform well, lack of clear conduct agreements, easy access to technology might, cultural differences in academic norms, and the fast-paced AI advance.

Universities should emphasize and clarify expected standards to ensure students develop critical thinking and communication skills essential for their future careers, while upholding ethical conduct and respecting intellectual property. Instructors may take a positive stance when

Oecologia Australis (ISSN: 2177-6199)
Ahead of print (https://revistas.ufrj.br/index.php/oa/issue/view/1109/showToc)
Article ID: AO#64848
Published online: 17 FEBRUARY 2026

it comes to using AI in teaching, built on the premise that AI tools are here to stay. For instance, generative AI can be used as a code assistant in some tutorials, with in-class discussions of the students' experience (Table S4).

Additional resources

There is a wide range of open and high-quality content to support and simplify teaching complex themes and processes in statistics and how R functions work. We listed main books and resources in Table S2 and Table S3, respectively. We also compiled a list of assessment types and applications for ecology courses using R (Table S4).

Final remarks

Teaching and learning R has the potential to massively contribute to developing professional skills, becoming a game changer in an ecologist's career (we can testify to this ourselves). Thus, we invite educators to join us on the journey of introducing ecology students to the "command line mindset". R has an engaged international community that enriches educational experience and makes this journey much easier.

As in any journey, though, teaching R to ecology students presents both significant challenges and opportunities. Students often face a steep learning curve, grappling with R syntax, and instructors must make use of a range of tailored resources and engaging projects to help them overcome these hurdles. Yet, mastering R opens many opportunities for ecology students in academia, industry, and government.

Proficiency in R enables students to take full advantage of these opportunities by addressing ecological questions with precision and transparency, and by enhancing their competitiveness in the job market. Thus, while the path to learning R may be challenging, the lasting benefits for ecology students are substantial, not only enhancing their analytical skills but also contributing to the collective growth of the global R community and of Ecology as a scientific discipline.

ACKNOWLEDGMENTS

We thank the worldwide communities of R and Stack Overflow for always helping us when we are stuck ourselves. Our undergraduate and graduate teaching assistants, with their fresh view on the fears and desires of younger generations, have also played a vital role in helping us learn how to teach R in the classroom. Finally, we are deeply indebted to the shiny, enthusiastic students we find in every class, whose curiosity and appetite for knowledge motivate us to improve our teaching a little more every day.

REFERENCES

Auker, L. A., & Barthelmess, E. L. 2020. Teaching R in the undergraduate ecology classroom: approaches, lessons learned, and recommendations. Ecosphere, 11(4). DOI: 10.1002/ecs2.3060

Bellamy, R. K. E. 1994. What Does Pseudo-Code Do? A Psychological Analysis of the use of Pseudo-Code by Experienced Programmers. Human-Computer Interaction.

Braga, P. H. P., Hébert, K., Hudgins, E. J., Scott, E. R., Edwards, B. P. M., Sánchez Reyes, L. L., Grainger, M. J., Foroughirad, V., Hillemann, F., Binley, A. D., Brookson, C. B., Gaynor, K. M., Shafiei Sabet, S., Güncan, A., Weierbach, H., Gomes, D. G. E., & Crystal-Ornelas, R. 2023. Not just for programmers: How GitHub can accelerate collaborative and reproducible research in ecology and evolution. Methods in Ecology and Evolution, 14(6), 1364–1380. DOI: 10.1111/2041-210X.14108

Carscadden, K., & Martin, A. 2022. To Tidy or not When Teaching R Skills in Biology Classes. Version 5. International Journal of Higher Education, 11(5), 39. DOI: 10.5430/ijhe.v11n5p39

Çetinkaya-Rundel, M., Hardin, J., Baumer, B. S., McNamara, A., Horton, N. J., & Rundel, C. 2022. An educator's perspective of the tidyverse. Technology Innovations in Statistics Education, 14(1). DOI: 10.5070/T514154352

649 Chow, F., Calixto, C. P. G., & Mello, M. A. R. 2021. Do ensino remoto emergencial ao ensino
650 híbrido no curso de Ciências Biológicas: a nossa visão a partir do Instituto de Biociências
651 da Universidade de São Paulo (IB-USP). Version Supl 1. Medicina (Ribeirão Preto),
652 54(Supl 1), e-185554. DOI: 10.11606/issn.2176-7262.rmrp.2021.185554
653 Cooper, N., Clark, A. T., Lecomte, N., Qiao, H., & Ellison, A. M. 2024. Harnessing large
654 language models for coding, teaching and inclusion to empower research in ecology and
655 evolution. Methods in Ecology and Evolution, n/a(n/a). DOI: 10.1111/2041-210X.14325
656 Cooper, N., & Hsing, P.-Y. (Eds.). 2017. A guide to reproducible code in ecology and
657 evolution.British Ecological Society.
658 Da Silva, F. R., Gonçalves-Souza, T., Paterno, G. B., Provete, D. B., & Vancine, M. H. 2022.
659 Análises Ecológicas no R. PE: NUPEEA: p. 640.
660 Deslauriers, L., McCarty, L. S., Miller, K., Callaghan, K., & Kestin, G. 2019. Measuring actual
661 learning versus feeling of learning in response to being actively engaged in the classroom.
662 Proceedings of the National Academy of Sciences, 116(39), 19251–19257. DOI:
663 10.1073/pnas.1821936116
664 Dormann, C. F., & Mello, M. A. R. 2023. Why we need a Canonical Ecology Curriculum. Basic
665 and Applied Ecology, 71, 98–109. DOI: 10.1016/j.baae.2023.05.009
666 Farrell, K. J., & Carey, C. C. 2018. Power, pitfalls, and potential for integrating computational
667 literacy into undergraduate ecology courses. Ecology and Evolution, 8(16), 7744–7751.
668 DOI: 10.1002/ece3.4363
669 Figueiredo, L., Scherer, C., & Cabral, J. S. 2022. A simple kit to use computational notebooks for
670 more openness, reproducibility, and productivity in research. PLOS Computational
671 Biology, 18(9), e1010356. DOI: 10.1371/journal.pcbi.1010356
672 Forrester, C., Schwikert, S., Foster, J., & Corwin, L. 2022. Undergraduate R Programming
673 Anxiety in Ecology: Persistent Gender Gaps and Coping Strategies. CBE—Life Sciences
674 Education, 21(2), ar29. DOI: 10.1187/cbe.21-05-0133

675 Foulkes, A. S. 2009. Applied Statistical Genetics with R: For Population-based Association
676 Studies. New York, NY: Springer New York. DOI: 10.1007/978-0-387-89554-3
677 Harrison, K. (Ed.). 2014. A guide to data management in ecology and evolution.British Ecological
678 Society.
679 Hoffman, A. M., & Wright, C. 2024. Ten simple rules for teaching an introduction to R. PLOS
680 Computational Biology, 20(5), e1012018. DOI: 10.1371/journal.pcbi.1012018
681 Kass, J. M., Vilela, B., Aiello-Lammens, M. E., Muscarella, R., Merow, C., & Anderson, R. P.
682 2018. Wallace: A flexible platform for reproducible modeling of species niches and
683 distributions built for community expansion. Methods in Ecology and Evolution, 9(4),
684 1151–1156. DOI: 10.1111/2041-210X.12945
685 Kruger, J., & Dunning, D. 1999. Unskilled and unaware of it: How difficulties in recognizing
686 one's own incompetence lead to inflated self-assessments. Journal of Personality and
687 Social Psychology, 77(6), 1121–1134. DOI: 10.1037/0022-3514.77.6.1121
688 Kuhn, M., & Wickham, H. 2020. Tidymodels: a collection of packages for modeling and machine
689 learning using tidyverse principles. (Retrieved on from https://www.tidymodels.org).
690 Lai, J., Cui, D., Zhu, W., & Mao, L. 2023. The Use of R and R Packages in Biodiversity
691 Conservation Research. Version 12. Diversity, 15(12), 1202. DOI: 10.3390/d15121202
692 Lai, J., Lortie, C. J., Muenchen, R. A., Yang, J., & Ma, K. 2019. Evaluating the popularity of R
693 in ecology. Ecosphere, 10(1), e02567. DOI: 10.1002/ecs2.2567
694 Martin, R. C. 2011. The clean coder: a code of conduct for professional programmers. Upper
695 Saddle River, NJ: Prentice Hall: p. 210.
696 Martinez-Blasco, M., Serrano, V., Prior, F., & Cuadros, J. 2023. Analysis of an event study using
697 the Fama–French five-factor model: teaching approaches including spreadsheets and the R
698 programming language. Financial Innovation, 9(1), 76. DOI: 10.1186/s40854-023-00477-
699 3
700 McCartney, M. 2016. Mistakes as a pathway to learning. Science, 352(6290), 1186.3-1187. DOI:
701 10.1126/science.352.6290.1186-c

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. 2019. A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. IEEE Transactions on Education, 62(2), 77–90. DOI: 10.1109/TE.2018.2864133

Mello, M. A. R., & Muylaert, R. L. 2020. Network Science as a Framework for Bat Studies. In: 21. Network Science as a Framework for Bat Studies. pp. 373–390. University of Chicago Press.

Nordmann, E., McAleer, P., Toivo, W., Paterson, H., & DeBruine, L. M. 2022. Data Visualization Using R for Researchers Who Do Not Use R. Advances in Methods and Practices in Psychological Science, 5(2), 25152459221074654. DOI: 10.1177/25152459221074654

Petre, M., & Winder, R. 1988. Issues governing the suitability of programming languages for programming tasks. In: Proceedings of the Fourth Conference of the British Computer Society on People and computers IV. pp. 199–215. USA: Cambridge University Press.

R Core Team. 2024. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org.

RStudio Team. 2020. RStudio: Integrated Development Environment for R. http://www.rstudio.com.

SEAD-UFBA. 2020a. As condições para aprendizagem online dos estudantes de graduação da UFBA em tempos de pandemia da Covid-19.Superintendência de educação a distância da Universidade Federal da Bahia.

SEAD-UFBA. 2020b. Diagnóstico das competências digitais dos professores da UFBA.Superintendência de educação a distância da Universidade Federal da Bahia.

Selwyn, N. 2022. Education and technology: key issues and debates. Third edition Third edition ed. London New York Oxford New Delhi Sydney: Bloomsbury Academic: p. 222.

Touchon, J. C., & McCoy, M. W. 2016. The mismatch between current statistical practice and doctoral training in ecology. Ecosphere, 7(8), e01394. DOI: 10.1002/ecs2.1394

Wickham, H. 2014. Tidy Data. Journal of Statistical Software, 59(10). DOI: 10.18637/jss.v059.i10

Wickham, H. 2016. ggplot2: Elegant Graphics for Data Analysis. 2 ed.Springer International Publishing. DOI: 10.1007/978-3-319-24277-4

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T., Miller, E., Bache, S., Müller, K., Ooms, J., Robinson, D., Seidel, D., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., & Yutani, H. 2019. Welcome to the Tidyverse. Journal of Open Source Software, 4(43), 1686. DOI: 10.21105/joss.01686

Wickham, H., & Bryan, J. 2023. R Packages: Organize, Test, Document, and Share Your Code. 2º edição. O'Reilly Media.

Wickham, H. 2019. Advanced R. 2º edição. Boca Raton: Chapman and Hall/CRC: p. 588.

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. 2023. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition ed. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly Media: p. 576.

Wilkinson, L., Wills, D., Rope, D., Norton, A., & Dubbs, R. 2005. The Grammar of Graphics. 2nd 2005 ed. edição 2nd 2005 ed. edição ed. New York: Springer.

Wing, J. M. 2006. Computational thinking. Communications of the ACM, 49(3), 33–35. DOI: 10.1145/1118178.1118215

Wu, T.-T., Asmara, A., Huang, Y.-M., & Permata Hapsari, I. 2024. Identification of Problem-Solving Techniques in Computational Thinking Studies: Systematic Literature Review. Sage Open, 14(2), 21582440241249897. DOI: 10.1177/21582440241249897

Xie, Y., Allaire, J. J., & Grolemund, G. 2019. R Markdown: the definitive guide. Boca Raton: CRC Press, Taylor and Francis Group: p. 1.

Yilmaz, R., & Karaoglan Yilmaz, F. G. 2023. The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. Computers and Education: Artificial Intelligence, 4, 100147. DOI: 10.1016/j.caeai.2023.100147

Yin, B., & Shen, Y. 2024. Development and Validation of the Compensatory Belief Scale for the Internet Instant Gratification Behavior. Heliyon, 10(1). DOI: 10.1016/j.heliyon.2024.e23972

Zuur, A. F., & Ieno, E. N. 2016. A protocol for conducting and presenting results of regression-type analyses. Methods in Ecology and Evolution, 7(6), 636–645. DOI: 10.1111/2041-210X.12577

Zuur, A. F., Ieno, E. N., & Elphick, C. S. 2010. A protocol for data exploration to avoid common statistical problems. Methods in Ecology and Evolution, 1(1), 3–14. DOI: 10.1111/j.2041-210X.2009.00001.x

Submitted: 29 July 2024

Accepted: 26 January 2026

Published: 17 February 2026

Associate Editor: Dr. Marcus Vinícius Vieira