1 *Challenges of teaching the R programming language to ecologists*

2

THE CHALLENGES AND NUANCES OF TEACHING THE R

PROGRAMMING LANGUAGE TO ECOLOGISTS

5

6 *Maurício Humberto Vancine[1]* (iD), Pavel Dodonov[2] (iD), Bruno Vilela[2] (iD), Luisa Diele-Viegas[3] (iD)*

7 *, Victor Casagrande Souza[4] (iD), Felipe Alvarez Silva Nunes[4] (iD), Ana Julia Oliveira Silva[4] (iD),*

8 *Beatriz Milz[5] (iD), Cristina A. Kita[6] (iD), Marco A. R. Mello[6] (iD) & Renata L. Muylaert[7] (iD)*

9

10 [1] Universidade Estadual de Campinas (UNICAMP), Instituto de Biociências, Departamento de Biologia Animal, Rua Monteiro

11 Lobato, 255, Cidade Universitária Zeferino Vaz, Barão Geraldo, 13083-862, Campinas, SP, Brasil

12 [2] Universidade Federal da Bahia, Instituto de Biologia, Laboratório de Ecologia Espacial, Rua Barão de Jeremoabo, 147, Campus de

13 Ondina, Ondina, 40170-115, Salvador, Bahia, Brasil.

14 [3] Universidade Federal da Bahia, Instituto de Biologia, Laboratório de (Bio)Diversidade no Antropoceno, Rua Barão de Jeremoabo,

15 147, Campus de Ondina, Ondina, 40170-115, Salvador, Bahia, Brasil.

16 [4] Universidade Estadual Paulista (Unesp), Instituto de Biociências, Departamento de Biodiversidade, Avenida 24 A, 1515, Bela Vista,

17 13506-900, Rio Claro, SP, Brasil.

18 [5] Universidade de São Paulo, Instituto de Energia e Ambiente, Av. Prof. Luciano Gualberto, 1289, Cidade Universitária, Butantã,

19 05508-010, São Paulo, SP, Brasil.

20 [6] Universidade de São Paulo, Instituto de Biociências, Departamento de Ecologia, Rua do Matão, travessa 14, 321, Cidade

21 Universitária, Butantã, 05508-090, São Paulo, SP, Brasil.

22 [7] The University of Sydney, Faculty of Science, Sydney School of Veterinary Science, Disease Ecology Laboratory, Parramatta Road,

23 s/n, Camperdown, 2006, Sydney, New South Wales, Austrália.

24 E-mails: mauricio.vancine@gmail.com (*corresponding author); pdodonov@gmail.com; bruno.vilela@ufba.br;

25 luisa.mviegas@gmail.com; vicasagrand@outlook.com; felipe.alvarez@unesp.br; anajuoliveira019@gmail.com;

26 milz.bea@gmail.com; c.akemikita@gmail.com; marmello@usp.br; renata.muylaert@sydney.edu.au

27

28 **Abstract.** The R programming language, esteemed for its statistical prowess, data management,

29 data visualization, and machine learning capabilities, has become a cornerstone of data science

30 applied to Ecology. Its open-source nature and broad array of analytical tools have garnered a

31 user base, mainly among ecologists. Nevertheless, instructing R to biology and ecology students

32   presents some key challenges. Here, we delve into the nuances of teaching R in graduate and

33   undergraduate ecology courses, focusing on data wrangling, data visualization, and statistics. We

34   provide insights from educators at various career stages, with a strong focus on the Brazilian

35   context. No single way of teaching R fits all situations; however, some general guidelines can be

36   provided and followed. Before starting this educational journey, the foundations must be

37   addressed, including infrastructure, hardware, and software. Once these prerequisites are secured,

38   students confront the intricacies of R's programming landscape. Abstract concepts, coding

39   idiosyncrasies, package compatibility, and the interplay between code and data need to be

40   mastered. The narrative progresses to the challenge of interpreting R's outputs and integrating

41   them seamlessly into statistical and ecological analyses. Additionally, we consider the impact of

42   structural challenges and global pressures on R education, such as the COVID-19 pandemic and

43   the evolving influence of artificial intelligence tools. In conclusion, exploring R within ecology

44   envisions a future where professionals possess the analytical skills to unlock innovative solutions

45   and applications. As this educational journey concludes, we present our perspective on the future

46   of R teaching and how it can help develop our science.

47   **Keywords:** Brazil, coding, ecology, data science, higher education, programming, science

48   education.

49

50   **Resumo**. A linguagem de programação R, estimada por sua habilidade estatística, gerenciamento

51   de dados, visualização de dados e capacidades de aprendizado de máquina, tornou-se uma pedra

52   angular da ciência de dados aplicada à Ecologia. Sua natureza de código aberto e ampla gama de

53   ferramentas analíticas conquistaram uma base de usuários, principalmente entre os ecólogos. No

54   entanto, instruir R para estudantes de biologia e ecologia apresenta alguns desafios-chave. Aqui,

55   mergulhamos nas nuances de ensinar R em cursos de graduação e pós-graduação em ecologia,

56   focando na manipulação de dados, visualização de dados e estatísticas. Fornecemos percepções

57   de educadores em várias fases da carreira, com forte foco no contexto brasileiro. Não existe uma

58   única maneira de ensinar R que se adapte a todas as situações; no entanto, algumas diretrizes

59   gerais podem ser fornecidas e seguidas. Antes de iniciar essa jornada educacional, é necessário

60   abordar os fundamentos, incluindo infraestrutura, hardware e software. Uma vez garantidos esses

61   pré-requisitos, os estudantes enfrentam as complexidades do ambiente de programação do R.

62   Conceitos abstratos, idiossincrasias de codificação, compatibilidade de pacotes e a interação entre

63   código e dados precisam ser dominados. A narrativa progride para o desafio de interpretar os

64   resultados do R e integrá-los perfeitamente em análises estatísticas e ecológicas. Além disso,

65   consideramos o impacto dos desafios estruturais e as pressões globais na educação em R, como a

66   pandemia de COVID-19 e envolvendo a influência das ferramentas de inteligência artificial. Em

67   conclusão, explorar o R na ecologia vislumbra um futuro em que os profissionais possuem as

68   habilidades analíticas para desbloquear soluções e aplicações inovadoras. À medida que essa

69   jornada educacional termina, apresentamos nossa perspectiva sobre o futuro do ensino de R e

70   como ele pode ajudar a desenvolver nossa ciência.

71   **Palavras-chave**: Brasil, codificação, ecologia, ciência de dados, ensino superior, programação,

72   educação científica.

73

INTRODUCTION

75   Discussions about a canonical curriculum in Ecology (Dormann & Mello 2023) have

76   sparked debates on how to teach natural phenomena, problem-solving, and programming, while

77   also highlighting the need to update the data analysis methods taught in undergraduate and

78   graduate courses. Use of the R programming language (R Core Team 2024) for ecological data

79   analysis has been rising continuously in the past two decades (Touchon & McCoy 2016) and has

80   become an important skill for ecologists (Auker & Barthelmess 2020). Teaching R, however, may

81   be challenging, especially for students with little or no programming background (Medeiros *et al.*

82   2019, Auker & Barthelmess 2020).

83   Traditionally, ecological data analysis in Brazil mainly relied on proprietary software, in

84   most cases costly and inflexible. The turn towards programming began to change this scenario in

85   the early 2000s, when R 1.0.0 was released to the public (https://cran.r-

86   project.org/bin/windows/base/old). Brazilian institutions slowly began integrating R into their

87   curricula, although not systematically or uniformly. To facilitate the change, workshops, courses,

88   and online resources proliferated, making R learning tools accessible to students and professionals

89   worldwide. Today, R is a staple of ecological research (Foulkes 2009, Lai *et al.* 2019, Mello &

90   Muylaert 2020, Da Silva et al. 2022, Wickham *et al.* 2023), with its use growing also in the

91   classroom. Using R can empower new generations of ecologists to conduct creative, robust, and

92   reproducible research, fostering a more open, collaborative, and innovative scientific community.

93   Despite this game-changing potential, teaching R to ecology students presents many

94   challenges, largely shaped by student preconceptions about mathematics, statistics, and

95   programming (Auker & Barthelmess 2020). Many students enter ecology with a strong passion

96   for natural history, but also a considerable degree of anxiety related to quantitative skills. This

97   can create a mental hindrance, leading students to believe that proficiency in R and programming

98   is beyond their reach.

99   Another difficulty in teaching R can be described as the "point-and-click mindset" versus

100   the "command line mindset". The use of friendly graphical user interfaces (GUIs) fosters clicking

101   through menu items rather than understanding how the underlying code is built. While GUIs make

102   technology widely accessible, they limit deeper understanding (Selwyn 2022). This mindset

103   seems to be even more prevalent among younger generations due to their constant exposure to

104   GUIs on smartphones and tablets. Reliance on GUIs is reinforced by educational tools, software

105   trends, and a cultural emphasis on instant gratification (Yin & Shen 2024), and can lead to low

106   engagement in programming courses. Students may also find programming intimidating and

107   prefer sticking to more user-friendly tools (Farrell & Carey 2018). To address this issue,

108   educational programs are incorporating both graphical and command-line interfaces to improve

109   computational literacy, fomenting the transition to a "command line mindset", where students

110   write and understand code.

111   Ultimately, in our understanding, the key to teaching R to ecology students lies in

112   addressing their preconceptions and insecurities, providing them with a solid computer science

113   foundation and effective troubleshooting strategies. By doing so, instructors can empower

114   students to overcome their initial apprehension and develop confidence in programming. This not

115   only enhances their capacity to conduct sophisticated ecological research but also broadens their

116   skill set, making them more versatile scientists with higher employability. Through structured

117   guidance and supportive teaching practices, students can transform their initial trepidation into a

118   strong proficiency in R, leveraging its powerful capabilities to advance their careers. The

119   successful teaching of R to Ecology students hinges on a well-rounded approach that combines

120   solid foundations, advanced analytical skills, and active learning strategies. In the next sections,

121   we discuss some ways to achieve these goals. We intend our suggestions to be applicable to a

122   wide range of course levels, from introductory undergraduate courses to students without previous

123   contact with R or statistics, to more advanced Masters and PhD courses.

124

125   *Increasing computational literacy*

126   To help students overcome these problems, instructors may emphasize the practical utility

127   of R in ecological research, demonstrating how it simplifies complex data analysis and

128   visualization, ultimately fomenting their analytical independence and making their scientific

129   inquiries more robust and insightful. This may be especially useful in introductory courses or for

130   students who didn't have previous contact with R or programming. Instructors could also show

131   students that programming literacy is critical to understanding the ecological literature, as most

132   analyses, including methods, are currently coded in R (Zuur et al. 2010, Zuur & Ieno 2016). To

133   succeed as professional ecologists, it is imperative for students to understand how data are

134   analyzed in key scientific papers.

135   A foundational understanding of computer science is also essential for learning R

136   effectively. This includes familiarity with concepts such as operating systems, file types, directory

137   structures, data types, control structures (e.g., loops and conditionals), language vocabulary and

138   syntax, functions and arguments, and the logic of how computers execute code. Specifically for

139   R, this also includes understanding functions and packages. Functions are pieces of code that

140 perform a specific task, with a structure like *function_name(argument1, argument2, ...)*. We can

141 draw parallels to the concept of a mathematical function *f(x)*: "f" is the function that performs an

142 operation to modify x; "x" represents the input data or arguments. Packages are a collection of R

143 functions, sample data, and documentation that describes how to use them (Wickham & Bryan

144 2023). They contain functions designed to perform specific tasks beyond those installed in Base

145 R. There are thousands of packages (discover for yourself in R: `nrow(available.packages())` or

146 at this link: https://cran.r-project.org/web/packages) for a wide range of purposes; the most

147 popular R packages among ecologists include *lme4*, *vegan*, *nmle*, *ape*, and *MuMIn* (Lai *et al.*

148 2023). Packages can be available as revised versions on the *Comprehensive R Archive Network*

149 (CRAN), or as development versions on GitHub and other versioning platforms.

150 Given that most beginner ecology students (e.g., undergraduates taking their first ecology

151 courses), as well as some more advanced ones (e.g., grad students who had not had previous

152 contact with R), need to gain such a background (Braga *et al.* 2023), instructors should start with

153 these fundamentals before delving into more complex R particularities. Introducing these

154 concepts through relatable ecological examples can bridge the gap between theoretical knowledge

155 and practical application, fostering a clearer understanding of programming logic.

156 Programming also involves dealing with bugs and errors, meaning that students must

157 develop a systematic approach to debugging and troubleshooting. Encouraging a "command line

158 mindset" that views errors not as failures but as learning opportunities can significantly enhance

159 their problem-solving skills (Kruger & Dunning 1999). Instructors should teach students how to

160 interpret error messages, use debugging tools, and employ strategies such as code commenting

161 and incremental testing to identify and fix issues (Zuur *et al.* 2010). Collaborative learning

162 environments where students can share and discuss shared challenges provide additional support

163 and reduce debugging frustration (McCartney 2016).

164

165 *Types of courses*

166   Before teaching R, it is important to define the type of course being taught: is the course

167   focused on R itself (e.g., an R programming course), or does it use R as a tool to teach other

168   subjects (e.g., a statistics course that uses R, but that could also use "point-and-click" software)?

169   We can also have a middle ground – a course aimed at teaching the concepts and their applications

170   in the R environment, in which learning R is an essential part of the course, but the course itself

171   is not limited to it. Thus, it is essential to clarify how R fits into the course—is it the subject or a

172   tool?—and to communicate this to the students early on. It is also essential to have clear objectives

173   in mind and an assessment of the infrastructure available, both at the teaching institution and at

174   students' homes. For instance, not all universities have computer labs that students can use at will,

175   which enormously facilitate learning R, especially for students who come from low-income

176   families.

177   Teaching the R programming language to ecology students also requires a structured

178   approach that balances basic and advanced courses while incorporating active methodologies to

179   enhance learning outcomes (Deslauriers *et al.* 2019, Hoffman & Wright 2024). To succeed in

180   learning R, students must be prepared to engage deeply with both the conceptual and practical

181   aspects of the language. Basic courses focused on R should lay a strong foundation by introducing

182   students to fundamental programming concepts and R syntax (see, for instance, "Use of R

183   Language for Data Analysis": https://uspdigital.usp.br/janus/Disciplina?sgldis=BIE5782& and

184   "R Programming": https://www.coursera.org/learn/r-programming). These courses typically

185   cover data types, the logic of functions, their parameters, and arguments, data manipulation, and

186   basic R visualizations. Abstract concepts should be tied to ecological data and research questions,

187   for example, small empirical datasets can be used for creating objects and applying functions to

188   them (an example used by one of us consists of ten values of bivalve cover associated to ten values

189   representing water quality, collected in the same city where the university is located, at sites that

190   are known to most of the students). Other data already organized as Data Papers (e.g., ATLANTIC

191   SERIES   -   https://esajournals.onlinelibrary.wiley.com/doi/toc/10.1002/(ISSN)1939-

192   9170.AtlanticPapers), can also be used, mainly because they are usually extensive (there are many

193   rows and columns), may need to join different tables, and almost always have missing data and

194   even typing errors that can be interesting data manipulation challenges for students. Hands-on

195   practice, with regular assignments involving real-world datasets, is also crucial.

196   Students in more advanced R courses (see, for instance, "Regression Modeling in Practice":

197   https://www.coursera.org/learn/regression-modeling-practice), should encounter more complex

198   topics such as advanced statistical methods, modeling, and the writing of user-defined functions.

199   These courses demand a higher level of engagement and problem-solving skills. To succeed,

200   students must be proficient in the basics and ready to tackle complex analyses. Advanced courses

201   should emphasize the application of R to exciting ecological problems, encouraging students to

202   develop customized scripts and functions. The ability to independently troubleshoot and optimize

203   code is critical at this stage, as is the capacity to interpret and communicate complex results

204   effectively. More advanced courses can include writing packages, which require advanced

205   knowledge of functions and their descriptions.

206   Conversely, a course that uses R as a tool and not as the subject may emphasize a small set

207   of basic skills and functions, enabling students to take a first step in learning R. These skills may

208   include reading spreadsheets in R, basic descriptive statistics, data visualization, and simple

209   statistical tests, focusing on real-world data and applications. Thus, even without gaining detailed

210   knowledge of R syntax and data structures, the student may be able to use R for basic analysis.

211   Still, some understanding of the underlying language (e.g., what are objects and functions) is

212   needed, and, if possible, teaching the creation of functions from scratch can be highly beneficial.

213   In addition to basic and advanced R courses, this programming language can also help

214   teach truly ecological courses. For instance, at the University of São Paulo, Brazil, there are labs

215   about animal ecology written as R notebooks in an in-person undergraduate course (see

216   "Advanced   Topics   in   Animal   Ecology":

217   https://uspdigital.usp.br/jupiterweb/obterDisciplina?nomdis=&sgldis=bie0315). In these labs,

218   students reproduce data analyses found in ecological papers and are expected to link concepts and

219   analyses through a combination of lectures, readings, quizzes, and labs. The same strategy is used

220   by us in a remote course on network science, which also includes R notebooks in its labs (see

221   "Ecological Networks": https://www.coursera.org/learn/redesecologicas).

222   Active methodologies play a pivotal role in teaching all those courses. These approaches,

223   which include flipped classroom, peer learning, and project-based learning, require students to be

224   protagonists in the learning process (Deslauriers et al. 2019, and see proposal in Cap. 1 from Da

225   Silva et al. 2022 - https://analises-ecologicas.com/cap1#como-ensinar-e-aprender-com-esse-

226   livro). Students must be willing to engage proactively, work collaboratively, and embrace the

227   iterative nature of coding. We realize that not all students have this level of motivation; however,

228   in our experience, classes with hands-on experience, in which the students are able to solve

229   problems and see their code working, can boost engagement. Furthermore, keeping students

230   motivated to learn programming can be a significant challenge, as not everyone has a natural

231   aptitude or aptitude for logical and abstract concepts. Therefore, we believe that practical

232   examples and applications are good motivators for students to maintain a high level of interest.

233   In a flipped classroom, students might watch recorded lectures, read papers on their own,

234   and then engage in hands-on coding exercises during in-person class time. In project-based

235   learning, students may organize themselves into groups and work on developing a research project

236   or solving a problem on their own, with "scaffolding" provided by the instructors. The group

237   projects may be well-defined or more open, simulating real-world ecological research scenarios.

238   They may also be solved in a single lesson or span several weeks. By integrating these

239   methodologies, educators can create a dynamic and supportive learning environment that

240   encourages students to take ownership of their learning journey.

241   Finally, there may also be huge variation in learning speed and skill level among the

242   students, and strategies for maximizing the learning experience for all students, such as

243   standardizing classes (known in Portuguese as *aulas de nivelamento*), may increase the odds of

244   providing a uniform experience for everyone. Working on basic and advanced activities

245   encompassing different skill levels in tandem is also possible. In addition, more skilled students,

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

247   their colleagues.

248

249   *Teaching computational thinking*

250   One of the most significant challenges students face when learning R is not mastering the

251   language's syntax (we all know that even advanced users resort to search engines (e.g., Google)

252   or generative artificial intelligence (e.g., ChatGPT) regularly) but understanding how to approach

253   complex problems (Martinez-Blasco *et al.* 2023). Students often freeze at the initial stages

254   because they attempt to address the entire problem simultaneously and do not know where to

255   begin. The key issue is operationalizing the issue into smaller difficulties that are easier to solve,

256   a process also called "computational thinking" (Wing 2006). We suggest a simple, practical

257   structure adapted to help students break R problems into smaller, more manageable parts (see Wu

258   *et al.* 2024 for general propositions): (1) define the problem; (2) define the input and output; (3)

259   break down the processing steps; and (4) translate the steps into code.

260   *Define the problem:* The first step is to define the problem itself and operationalize it. This

261   can be straightforward, such as creating a function to calculate species richness in multiple

262   locations, or more abstract, such as describing the biodiversity of several protected areas. In the

263   latter, we need to clarify the biodiversity metrics to be utilized before attempting to address the

264   question itself. Therefore, we must operationalize our variables (clearly define and specify the

265   proxies for abstract concepts), just as we do in hypothesis testing. This is a skill that can be

266   explored from beginner to advanced courses, with problems of different levels.

267   *Define the input and output:* The next step is to determine the input data, or information,

268   that should be provided by the user to solve the problem. For example, if we want to calculate the

269   species richness of two or more locations, we need a community matrix, where rows represent

270   locations and columns represent species, and each value denotes the abundance or presence of a

271   species at a given location. We then define the output, which should be defined now, as one should

272   be able to clearly foresee the structure of their result before obtaining them. This enables

273   individuals to align their efforts with their ultimate objectives, fostering efficiency and

274   effectiveness in problem-solving. In our example, the expected output is species richness per

275   location. Defining the input and output beforehand also allows any code to be modular, composed

276   of independent, reusable units or modules, each with a specific function, which increases

277   efficiency and collaboration in science (e.g., Kass 2018).

278   *Break down the processing steps:* We can now move forward to breaking down the problem

279   into processing steps. Instead of asking students to calculate species richness, we need to guide

280   them on seeing the larger task as a sequence of smaller, manageable steps. One approach is to

281   provide them with a simplified version of their input data and ask them to calculate species

282   richness by hand, paying attention to the mental steps used in the calculation and writing them

283   down, including input and output information. This may result in something like:

284

285   1. Input data: Community matrix.

286   2. Identify the species with abundance values greater than 0 at a given location.

287   3. Count the number of unique species identified in step 1.

288   4. Write/store this number.

289   5. Repeat this process for all locations.

290   6. Output data: Species richness values per location.

291

292   These written steps to perform a task are also known as a pseudocode: a simplified,

293   informal protocol written in natural language, which is used to outline the logic and steps of an

294   algorithm in a more friendly way (Petre & Winder 1988). It is not tied to any specific

295   programming language and avoids complex syntax, useful for planning and communicating how

296   a script or function will work before writing the actual code (Bellamy 1994).

297   *Translate the steps into code:* Now that the problem of creating a function to calculate

298   species richness across multiple locations has been largely decomposed into manageable steps,

299   its pseudocode can be translated into actual R code. The pseudocode can also serve as

300 documentation, enhancing the code's clarity and reproducibility. Students can then test the

301 function, optimize its performance and even explore entirely different algorithms. For instance,

302 students could improve the algorithm by taking vectorization and avoiding step 5, which requires

303 a more complex structure involving a loop (a control structure for a set of instructions to be

304 executed repeatedly over a sequence of values). This structured approach also equips students

305 with essential problem-solving skills, such as abstract thinking, systematic planning, and iterative

306 refinement, which are invaluable for tackling a wide range of challenges they may encounter in

307 their careers.

308

309 *Online teaching*

310   The social isolation caused by lockdowns during the COVID-19 pandemic has greatly

311 impacted higher education, forcing a switch from in-person to online and then hybrid teaching in

312 universities worldwide (Chow *et al.* 2021). After the pandemic, many teaching activities

313 continued remotely or semi-remotely, often using the experience gained during the pandemic.

314 Instructors who had limited experience with digital teaching resources (SEAD-UFBA 2020b) had

315 to convert in-person courses to full remote versions overnight. Below, we provide some

316 recommendations for such courses based on our experience with online teaching before, during,

317 and after COVID-19.

318   The first step is to assess which tools are available for which students. Not all students have

319 access to computers at home. For example, a survey performed by the Federal University of Bahia

320 in 2020 (SEAD-UFBA 2020a) showed that approximately a quarter of its students had little to no

321 conditions for online learning, only 18% had access to a desktop personal computer and 72% to

322 a notebook, and approximately 1% did not have access to any digital device. Internet access was

323 limited for approximately 20% of the students. These data provide a glimpse of the dire situation

324 faced by many students, especially in developing countries. Although the conditions vary widely

325 among universities, educators must understand that not all students have access to a computer

326 with fast, stable internet, and those with access might need to share the device with other people

327 in their household or neighborhood. Thus, for undergraduate and graduate courses in vulnerable

328 communities (see **Structural challenges)**, we recommend first performing an institutional survey

329 with the students to assess what infrastructure they have available before choosing learning

330 strategies and tools. For outreach courses, an alternative would be to list the equipment required.

331 It is also important to keep in mind that some packages may be too memory-consuming for older

332 computers, and the minimal requirements for running the course should be mentioned in the

333 syllabus, including online alternatives such as Posit Cloud (formerly RStudio Cloud)

334 (https://posit.cloud).

335 In addition, most students will not have two screens available. If they must follow the

336 instructor's demonstration and write their own code at the same time, they are likely to become

337 lost along the way. One alternative is to separate teaching and coding/writing moments: the

338 instructor first makes a demonstration, and then the students are given time to work on their own

339 code, based on their notes. Providing preparation material in advance as PDF, R and R

340 markdown/quarto files with task checklists is highly recommended.

341 Another barrier is that whereas in an in-person course the instructor can move around the

342 classroom by checking the code, giving suggestions and correcting errors, this is challenging in

343 an online class, especially if the number of instructors and assistants is not enough to reach all

344 students. Commonly used online tools tend to lack interactivity and engagement, making it

345 difficult to get help or communicate issues due to limited attention (often the student is not

346 allowed to share their screen by default). Internet connection issues disrupt learning and frustrate

347 students and instructors. One way to deal with those challenges is to use smaller break rooms

348 where students can enter, share their screen with an instructor or assistant, and get help to solve

349 their issues. Another alternative is for students with difficulties to share their screen with the entire

350 class, so everyone, including the instructor, can help solve their issues. This latter alternative can

351 be particularly interesting, as by helping others fix their colleagues' issues, students can improve

352 their own skills.

353   Group activities enable the participation of students having access only to smartphones or

354   tablets, as we observed in our courses during the pandemic. Instead of each student writing their

355   own code, they can be divided into smaller groups, in which one student shares their screen with

356   the group and everyone contributes. The instructor may interact with each group for a short time

357   to guide the students. This last approach is especially interesting for active strategies, such as

358   project-based learning. After a brief explanation, the students may be given a problem to solve in

359   R, such as analyzing a given dataset to answer a question. The students then divide themselves

360   into groups, each group using a different online room or a separate call. While each group works

361   on the problem, the instructor interacts with each group in question, and at the end of the lesson,

362   all groups meet and share their solutions.

363   Finally, it is important to have a well-organized list of online resources, such as texts and

364   video-lectures, for students to be able to keep learning beyond class time. This is valid also for

365   in-person teaching, but even more so for synchronous online courses (see **Structural problems**).

366   Fully asynchronous courses are also an option, especially for outreach. For these courses, we

367   recommend using a combination of videos, texts, quizzes, and labs for the students to solve,

368   individually or in groups.

369

370   *Tutoring*

371   Since many students have limited or no programming experience, having guest lecturers

372   and teaching assistants to aid the main instructor can greatly improve the learning process. First,

373   students might feel more comfortable asking questions to a tutor than to a professor, because

374   tutors may offer personalized attention and a more approachable environment, especially in

375   larger, more formal class settings led by professors. Even in smaller classes, with 15–20 students,

376   it may be impossible for a single instructor to give attention to all students. In these cases, tutors

377   can offer office hours to individual students or groups and stimulate peer-learning sessions,

378   enhancing the students' understanding while also fostering a sense of community and teamwork.

379   In addition, beginners may struggle with basic R syntax, such as missing parentheses, quotation

380   marks, or incorrect function names. R error messages can also be cryptic at first, making

381   debugging challenging, time-consuming, and frustrating. By quickly identifying syntax problems,

382   tutors can optimize learning.

383   For students with little knowledge of statistics and experimental/sampling design,

384   understanding and applying statistical concepts to data analysis is challenging. However, the

385   hands-on experience with R can also aid in learning these concepts, especially if this experience

386   is guided by tutors. Tutors can assist students by offering help sessions on concepts and

387   applications through data analysis and help students understand what has been asked in their

388   assignments. If needed, tutors can also encourage good data management practices (Broman &

389   Woo 2018) during these sessions, such as the principles of tidy data (Wickham 2014), to make

390   their sheets organized and human- and machine-readable (Harrison 2014, Cooper & Hsing 2017).

391   Tutors can also help with the biological interpretation of statistical model results.

392   Although there are many resources available to learn R on the internet, the quality, and

393   accessibility of these resources varies largely. Tutors can guide students by indicating good

394   sources of information, including books, online forums, and online courses (see Supplementary

395   Material). The exact nature of this contribution may be something like providing a pseudocode

396   for the students to apply; aiding the students' reasoning by asking guiding questions; help with

397   resolving R-specific issues; or something else, depending on the course's nature.

398   Finally, on a related topic, peer-learning outside the classroom is also great. An example is

399   the GERE (*Grupo de Estudos em R e Estatísticas*) [https://github.com/grupo-estudos-r-estatistica]

400   from Unesp – Rio Claro, a study group in statistics and R that was established in 2024 in response

401   to strong resistance to mathematical thinking in biological undergraduate courses. Initially

402   supported by a few members, the group's attendance grew significantly through social media and

403   word of mouth, reaching an average of twenty participants per meeting. Guided by the book

404   "Ecological Analysis in R" (Da Silva *et al.* 2022), the group conducted hands-on sessions twice

405   a week, at two times of the day, morning and evening, with the same content, covering topics

406   from basic R to linear model analysis, and encouraged sharing of research projects.

407

408 *Structural challenges*

409   Teaching R programming to ecology students in Brazil and other geopolitically peripheral

410 countries may be hindered by lengthy structural challenges stemming from a range of

411 infrastructural issues, political and economic instability, and logistical constraints. For instance,

412 universities in Brazil commonly struggle with intermittent internet connection, which disrupts

413 programming classes, especially if package updates and data download are required. This also

414 limits synchronous online courses, in which, due to internet instability, power shortages, and other

415 unpredictable issues, students may be unable to attend some lessons.

416   Another problem is outdated or limited computer labs (including limited number of chairs

417 and desks) and insufficient technical support. Many computer labs are equipped with outdated

418 machines that cannot efficiently run modern software, including the latest versions of R and its

419 multitude of packages. This forces students and instructors to spend valuable class time

420 troubleshooting technical issues rather than focusing on programming concepts and applications.

421   Lack of adequate IT support (technical help with computer systems) is also a challenge, as

422 dedicated personnel may be understaffed or unavailable, meaning that technical issues cannot be

423 resolved promptly, leading to delays and interruptions during classes. Moreover, instructors often

424 do not have administrative privileges to update or install software on university computers,

425 limiting their possibilities of troubleshooting. This is particularly problematic when software

426 updates or new package installations are required to run certain R scripts. Setting up programming

427 environments, troubleshooting software issues, and ensuring that all students can access the

428 necessary tools thus become formidable tasks.

429   The political instability in Brazil (see

430 https://dynamicecology.wordpress.com/2017/11/28/guest-post-doing-ecology-on-a-

431 rollercoaster-in-brazil for a perspective on the topic) frequently leads to strikes and protests by

432 both faculty and students – often at the end of academic semesters, when advanced topics in R

433 programming are typically covered. Because of this, crucial portions of the curriculum may be

434   rushed or superficially covered. Faculty and student strikes can also delay grading and feedback.

435   Without timely feedback, students may struggle to keep up with course requirements and fall

436   behind in their understanding of complex programming concepts. Sometimes, entire chunks of

437   the curriculum are not delivered at all due to time constraints.

438   Finally, logistical issues, including transportation and safety, impact student attendance and

439   participation, especially for nocturnal courses. Many students face challenges commuting to

440   campus due to unreliable public transportation and safety concerns (in extreme cases, safety

441   concerns lead to classes being suspended - https://www.correio24horas.com.br/minha-

442   bahia/cursos-na-ufba-suspendem-aulas-em-meio-a-violencia-no-alto-das-pombas-e-calabar-

443   0923). These issues result in irregular attendance, causing students to miss critical classes and

444   disrupting the continuity of their learning. The lack of continuity in attendance means that students

445   often miss important lectures and hands-on sessions. In a subject like R programming, where

446   concepts build on each other, missing even a few classes can significantly hinder a student's

447   performance.

448   Addressing these challenges requires consistent efforts to improve and maintain

449   infrastructure, provide reliable IT support, and mitigate logistical and safety concerns in campus

450   and public transportation, as well as great planning skills and adaptability on the instructors' part.

451

452   *Diversity and inclusion*

453   We view inclusion and equity as a continuous process involving structures, systems,

454   guidelines, and our teaching practice. Applying the equity and inclusion principles in our teaching

455   practice can vary according to institutional structure and systems, and guidelines for the tertiary

456   sector may vary in terms of curriculum delivery and assessments (Table S4). Acknowledging the

457   multiculturalism and extensive diversity of students in Brazil is a step forward in inclusion, and

458   advancing efforts to disseminate equitable practices should be at the forefront of developing

459   guidelines for teaching practice in Brazil. Barriers to computational literacy are grounded in

460   overall literacy inequality in Brazil, and they extend to language barriers, as much material for R

461 programming and most function descriptions are in English. Moreover, gender disparities are

462 common in undergraduate programming (Forrester *et al.* 2022).

463   A valuable initiative tackling such issues is R-ladies (https://rladies.org), a global

464 organization aiming to promote gender diversity in the R community. The organization was

465 founded in 2012 by Gabriela de Queiroz—a Brazilian woman studying in the US at that time. The

466 organization has since grown and there are chapters (local groups) in all continents, including

467 active chapters in Brazil, such as R-Ladies São Paulo (https://rladies-sp.org), R-Ladies Goiânia

468 (https://www.rladiesgyn.com), and R-Ladies Belo Horizonte (https://rladiesbh.com.br). The

469 chapters organize free activities, including online or in-person events, study groups, and book

470 clubs, focused on teaching R. The chapters also provide free online resources including

471 comprehensive tutorials, coding exercises, and recorded workshops on a wide range of topics,

472 from basic R programming to spatial data analysis. Additionally, R-Ladies maintain active blogs

473 and social media channels where members promote inclusion and diversity and share their

474 knowledge, experiences, and updates on the latest developments in the R community.

475

476 *Additional considerations and resources*

477 *IDEs e text editors*

478   The R programming language comes installed with a command line console, called RGui

479 on Windows, and directly on the terminal on Unix operational systems (GNU/Linux and macOS).

480 However, there are other ways to write R code through Integrated Development Environments

481 (IDEs), which assist with code completion, automatic code generation, error detection, and syntax

482 highlighting. Some IDEs have been specifically developed for R, such as RStudio (RStudio Team

483 2020), which combines an R console, a syntax-highlighting editor with an autocomplete function,

484 and tabs for plots, history, git, and help. RStudio may be taught alongside R, but it is important

485 to distinguish between the two. Recently, Posit is working on a next-generation data science IDE

486 called Positron™ (https://github.com/posit-dev/positron), which shifted to a language-agnostic

487 IDE built by Posit (formerly RStudio). Other IDEs are shown in Table S1.

488

489   *tidyverse and tidymodels*

490   R has undergone adaptations in recent years, including a new approach to data

491   manipulation, called Tidy Data (Wickham 2014), which focuses on data cleaning and

492   organization. Building on these ideas, the tidyverse (https://www.tidyverse.org) was implemented

493   in R as a collection of packages (e.g., *dplyr, ggplot2, magrittr, readr*, *tibble*, and *tidyr*) designed

494   for data importation, manipulation, exploration, visualization, analysis, and communication

495   (Wickham *et al.* 2019). This set of packages significantly alters data flow operations in R,

496   introducing the concept of piping (*%>%*) and functional programming (e.g., *purrr*, Wickham

497   2019). Following the same approach, *tidymodels* (Kuhn & Wickham 2020)

498   (https://www.tidymodels.org) is a collection of packages for modeling and machine learning.

499   There are questions on whether these modifications are here to stay or if it might be

500   beneficial to revert some aspects to the Base R version (Carscadden & Martin 2022, Çetinkaya-

501   Rundel et al. 2022). This poses a significant question: should R be taught starting with Base R,

502   focusing on its functions and syntax, or is it more advantageous to begin with the *tidyverse*? We

503   believe there is no definitive answer to this question yet, but we suggest, if time and equipment

504   permit, starting with Base R and then proceeding with tidyverse (Da Silva *et al.* 2022). Many

505   concepts used in the *tidyverse* can be confusing if the fundamental classes and object types are

506   not explained in Base R first.

507

508   *ggplot2*

509   Within the *tidyverse* approach, *ggplot2* (Wickham 2016) is a widely used package for

510   creating graphics (Nordmann *et al.* 2022). Based on the grammar of graphics (Wilkinson *et al.*

511   2005), *ggplot2* package has its own syntax based on layers, requiring specific functions which are

512   connected using the + operator. The dichotomy between Base R and the tidyverse also arises here,

513   with teaching challenges related to the plotting functions in Base R and the syntax and functions

514   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

542 used both for teaching and preparing teaching material, and as a way for students to integrate text

543 and code to build their own study material.

544

545 *Version control (git and GitHub)*

546 Version control is a system that tracks changes to files and directories over time, enabling

547 multiple users to collaborate on projects (Braga *et al.* 2023). Git (https://www.git-scm.com) is the

548 most common software for managing version control and supports integration with remote

549 repositories, such as GitHub (https://github.com); both are also integrated into RStudio. It tracks

550 changes, allowing for rollback, comparison, conflict resolution, and collaboration through

551 branches and mergers. Version control provides backup through commits and remote repositories,

552 facilitating peer review via pull requests. Teaching version control concepts is crucial to prepare

553 students for collaborative projects.

554

555 *Generative Artificial Intelligence (AI) tools*

556 The rise of large language model-based chatbots (generative AI) from late 2022 on (e.g.,

557 https://openai.com/index/chatgpt) marked a transformational milestone in computational tools

558 and natural language understanding, massively influencing various domains, including education

559 (Cooper *et al.* 2024). Generative AI models quickly create and review content (computer code,

560 narrative text, image, film, and audio), including data insights and complex code workflows, from

561 iterative prompts provided by the user, based on the patterns the tool has learned from and the

562 data it was trained on. Using generative tools in teaching R for students can enhance the learning

563 experience by providing personalized, real-time assistance and explanations (Yilmaz &

564 Karaoglan Yilmaz 2023). These tools can help students grasp complex concepts and syntax in R

565 by offering immediate feedback, logical explanations, cross-language examples (such as

566 translating pseudocode to R and then to Python) and troubleshooting advice. They can also build

567 workflows to tackle problems relevant to ecological studies. All this can foster a deeper

568 understanding of R, streamline the learning process, and help empower students to apply their

569 coding skills to real-world ecological research.

570 Nevertheless, using generative AI for teaching can lead to problems such as over-reliance

571 on AI (compromising critical thinking and independent problem-solving skills), concerns about

572 academic integrity through potential plagiarism, reduced engagement with foundational concepts,

573 decreased motivation for learning and retaining knowledge, and incorrect information, known as

574 hallucinations, provided by AI (see, for instance: https://revistapesquisa.fapesp.br/universidades-

575 brasileiras-discutem-regras-de-uso-de-inteligencia-artificial/). Unequal access to AI tools can

576 further exacerbate disparities in learning outcomes, while ethical questions persist regarding

577 content ownership and fairness in academic settings. Balancing the benefits of AI in education

578 requires addressing these challenges and generating clear institutional guidelines to ensure

579 responsible and effective use in the classroom.

580 Reactions to AI by university faculty vary from altogether banishing it to wholeheartedly

581 embracing   it   and   making   its   use   mandatory   in   assignments   (see

582 https://revistapesquisa.fapesp.br/pesquisadores-usam-inteligencia-artificial-em-tarefas-

583 academicas/). The discussions include academic integrity, ethical conduct, what the learning

584 experience is about; and what is expected from students in their assignments and learning

585 outcomes in face of a mutable job market. For students, this involves submitting their own original

586 work in assignments, properly acknowledging sources when using existing knowledge, adhering

587 to ethical guidelines in research, and following instructions. Breaches of academic integrity

588 undermine the trust in academic standards and the value of education. Still, upholding academic

589 integrity is challenging due to pressures to perform well, lack of clear conduct agreements, easy

590 access to technology might, cultural differences in academic norms, and the fast-paced AI

591 advance.

592 Universities should emphasize and clarify expected standards to ensure students develop

593 critical thinking and communication skills essential for their future careers, while upholding

594 ethical conduct and respecting intellectual property. Instructors may take a positive stance when

595   it comes to using AI in teaching, built on the premise that AI tools are here to stay. For instance,

596   generative AI can be used as a code assistant in some tutorials, with in-class discussions of the

597   students' experience (Table S4).

598

599   *Additional resources*

600   There is a wide range of open and high-quality content to support and simplify teaching

601   complex themes and processes in statistics and how R functions work. We listed main books and

602   resources in Table S2 and Table S3, respectively. We also compiled a list of assessment types and

603   applications for ecology courses using R (Table S4).

604

605   *Final remarks*

606   Teaching and learning R has the potential to massively contribute to developing

607   professional skills, becoming a game changer in an ecologist's career (we can testify to this

608   ourselves). Thus, we invite educators to join us on the journey of introducing ecology students to

609   the "command line mindset". R has an engaged international community that enriches educational

610   experience and makes this journey much easier.

611   As in any journey, though, teaching R to ecology students presents both significant

612   challenges and opportunities. Students often face a steep learning curve, grappling with R syntax,

613   and instructors must make use of a range of tailored resources and engaging projects to help them

614   overcome these hurdles. Yet, mastering R opens many opportunities for ecology students in

615   academia, industry, and government.

616   Proficiency in R enables students to take full advantage of these opportunities by

617   addressing ecological questions with precision and transparency, and by enhancing their

618   competitiveness in the job market. Thus, while the path to learning R may be challenging, the

619   lasting benefits for ecology students are substantial, not only enhancing their analytical skills but

620   also contributing to the collective growth of the global R community and of Ecology as a scientific

621   discipline.

622

630

631 **REFERENCES**

632 Auker, L. A., & Barthelmess, E. L. 2020. Teaching R in the undergraduate ecology classroom:

633   approaches, lessons learned, and recommendations. Ecosphere, 11(4). DOI:

634   10.1002/ecs2.3060

635 Bellamy, R. K. E. 1994. What Does Pseudo-Code Do? A Psychological Analysis of the use of

636   Pseudo-Code by Experienced Programmers. Human-Computer Interaction.

637 Braga, P. H. P., Hébert, K., Hudgins, E. J., Scott, E. R., Edwards, B. P. M., Sánchez Reyes, L. L.,

638   Grainger, M. J., Foroughirad, V., Hillemann, F., Binley, A. D., Brookson, C. B., Gaynor,

639   K. M., Shafiei Sabet, S., Güncan, A., Weierbach, H., Gomes, D. G. E., & Crystal-Ornelas,

640   R. 2023. Not just for programmers: How GitHub can accelerate collaborative and

641   reproducible research in ecology and evolution. Methods in Ecology and Evolution, 14(6),

642   1364–1380. DOI: 10.1111/2041-210X.14108

643 Carscadden, K., & Martin, A. 2022. To Tidy or not When Teaching R Skills in Biology Classes.

644   Version 5. International Journal of Higher Education, 11(5), 39. DOI:

645   10.5430/ijhe.v11n5p39

646 Çetinkaya-Rundel, M., Hardin, J., Baumer, B. S., McNamara, A., Horton, N. J., & Rundel, C.

647   2022. An educator's perspective of the tidyverse. Technology Innovations in Statistics

648   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

702 Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. 2019. A Systematic Literature Review on

703   Teaching and Learning Introductory Programming in Higher Education. IEEE

704   Transactions on Education, 62(2), 77–90. DOI: 10.1109/TE.2018.2864133

705 Mello, M. A. R., & Muylaert, R. L. 2020. Network Science as a Framework for Bat Studies. In:

706   21. Network Science as a Framework for Bat Studies. pp. 373–390. University of Chicago

707   Press.

708 Nordmann, E., McAleer, P., Toivo, W., Paterson, H., & DeBruine, L. M. 2022. Data Visualization

709   Using R for Researchers Who Do Not Use R. Advances in Methods and Practices in

710   Psychological Science, 5(2), 25152459221074654. DOI: 10.1177/25152459221074654

711 Petre, M., & Winder, R. 1988. Issues governing the suitability of programming languages for

712   programming tasks. In: Proceedings of the Fourth Conference of the British Computer

713   Society on People and computers IV. pp. 199–215. USA: Cambridge University Press.

714 R Core Team. 2024. R: A Language and Environment for Statistical Computing. R Foundation

715   for Statistical Computing, Vienna, Austria. https://www.R-project.org.

716 RStudio Team. 2020. RStudio: Integrated Development Environment for R.

717   http://www.rstudio.com.

718 SEAD-UFBA. 2020a. As condições para aprendizagem online dos estudantes de graduação da

719   UFBA em tempos de pandemia da Covid-19.Superintendência de educação a distância da

720   Universidade Federal da Bahia.

721 SEAD-UFBA. 2020b. Diagnóstico das competências digitais dos professores da

722   UFBA.Superintendência de educação a distância da Universidade Federal da Bahia.

723 Selwyn, N. 2022. Education and technology: key issues and debates. Third edition Third edition

724   ed. London New York Oxford New Delhi Sydney: Bloomsbury Academic: p. 222.

725 Touchon, J. C., & McCoy, M. W. 2016. The mismatch between current statistical practice and

726   doctoral training in ecology. Ecosphere, 7(8), e01394. DOI: 10.1002/ecs2.1394

727 Wickham, H. 2014. Tidy Data. Journal of Statistical Software, 59(10). DOI:

728   10.18637/jss.v059.i10

729   Wickham, H. 2016. ggplot2: Elegant Graphics for Data Analysis. 2 ed.Springer International

730   Publishing. DOI: 10.1007/978-3-319-24277-4

731   Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G.,

732   Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T., Miller, E., Bache, S., Müller, K.,

733   Ooms, J., Robinson, D., Seidel, D., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo,

734   K., & Yutani, H. 2019. Welcome to the Tidyverse. Journal of Open Source Software, 4(43),

735   1686. DOI: 10.21105/joss.01686

736   Wickham, H., & Bryan, J. 2023. R Packages: Organize, Test, Document, and Share Your Code.

737   2º edição. O'Reilly Media.

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

739   Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. 2023. R for Data Science: Import, Tidy,

740   Transform, Visualize, and Model Data. 2nd edition ed. Beijing Boston Farnham Sebastopol

741   Tokyo: O'Reilly Media: p. 576.

742   Wilkinson, L., Wills, D., Rope, D., Norton, A., & Dubbs, R. 2005. The Grammar of Graphics.

743   2nd 2005 ed. edição 2nd 2005 ed. edição ed. New York: Springer.

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

745   10.1145/1118178.1118215

746   Wu, T.-T., Asmara, A., Huang, Y.-M., & Permata Hapsari, I. 2024. Identification of Problem-

747   Solving Techniques in Computational Thinking Studies: Systematic Literature Review.

748   Sage Open, 14(2), 21582440241249897. DOI: 10.1177/21582440241249897

749   Xie, Y., Allaire, J. J., & Grolemund, G. 2019. R Markdown: the definitive guide. Boca Raton:

750   CRC Press, Taylor and Francis Group: p. 1.

751   Yilmaz, R., & Karaoglan Yilmaz, F. G. 2023. The effect of generative artificial intelligence (AI)-

752   based tool use on students' computational thinking skills, programming self-efficacy and

753   motivation. Computers and Education: Artificial Intelligence, 4, 100147. DOI:

754   10.1016/j.caeai.2023.100147

755 Yin, B., & Shen, Y. 2024. Development and Validation of the Compensatory Belief Scale for the

756 Internet Instant Gratification Behavior. Heliyon, 10(1). DOI:

757 10.1016/j.heliyon.2024.e23972

758 Zuur, A. F., & Ieno, E. N. 2016. A protocol for conducting and presenting results of regression-

759 type analyses. Methods in Ecology and Evolution, 7(6), 636–645. DOI: 10.1111/2041-

760 210X.12577

761 Zuur, A. F., Ieno, E. N., & Elphick, C. S. 2010. A protocol for data exploration to avoid common

762 statistical problems. Methods in Ecology and Evolution, 1(1), 3–14. DOI: 10.1111/j.2041-

763 210X.2009.00001.x

764

765

770

771

772

773