Received: 26 February 2024 | Accepted: 11 March 2024

DOI: 10.1111/2041-210X.14325

Methods in Ecology and Evolution

PERSPECTIVE

Harnessing large language models for coding, teaching and inclusion to empower research in ecology and evolution

Natalie Cooper[1] | Adam T. Clark[2] | Nicolas Lecomte[3] | Huijie Qiao[4] | Aaron M. Ellison[5,6]

[1]Science Group, Natural History Museum London, London, UK; [2]Department of Biology, University of Graz, Graz, Austria; [3]Canada Research Chair in Polar and Boreal Ecology and Centre d'Études Nordiques, Department of Biology, University of Moncton, Moncton, New Brunswick, Canada; [4]Key Laboratory of Animal Ecology and Conservation Biology, Institute of Zoology, Chinese Academy of Sciences, Beijing, China; [5]Harvard University Herbaria, Harvard University, Cambridge, Massachusetts, USA and [6]Sound Solutions for Sustainable Science, Boston, Massachusetts, USA

Correspondence
Natalie Cooper
Email: natalie.cooper@nhm.ac.uk

Handling Editor: Bob O'Hara

Abstract

1. Large language models (LLMs) are a type of artificial intelligence (AI) that can perform various natural language processing tasks. The adoption of LLMs has become increasingly prominent in scientific writing and analyses because of the availability of free applications such as ChatGPT. This increased use of LLMs not only raises concerns about academic integrity but also presents opportunities for the research community. Here we focus on the opportunities for using LLMs for coding in ecology and evolution. We discuss how LLMs can be used to generate, explain, comment, translate, debug, optimise and test code. We also highlight the importance of writing effective prompts and carefully evaluating the outputs of LLMs. In addition, we draft a possible road map for using such models inclusively and with integrity.
2. LLMs can accelerate the coding process, especially for unfamiliar tasks, and free up time for higher level tasks and creative thinking while increasing efficiency and creative output. LLMs also enhance inclusion by accommodating individuals without coding skills, with limited access to education in coding, or for whom English is not their primary written or spoken language. However, code generated by LLMs is of variable quality and has issues related to mathematics, logic, non-reproducibility and intellectual property; it can also include mistakes and approximations, especially in novel methods.
3. We highlight the benefits of using LLMs to teach and learn coding, and advocate for guiding students in the appropriate use of AI tools for coding. Despite the ability to assign many coding tasks to LLMs, we also reaffirm the continued importance of teaching coding skills for interpreting LLM-generated code and to develop critical thinking skills.

4. As editors of MEE, we support—to a limited extent—the transparent, accountable and acknowledged use of LLMs and other AI tools in publications. If LLMs or comparable AI tools (excluding commonly used aids like spell-checkers, Grammarly and Writefull) are used to produce the work described in a manuscript, there must be a clear statement to that effect in its Methods section, and the corresponding or senior author must take responsibility for any code (or text) generated by the AI platform.

KEYWORDS
artificial intelligence, ChatGPT, coding, inclusion, large language models, teaching

1 | INTRODUCTION

Artificial intelligence (AI) is a rapidly expanding field, with myriad uses in ecology and evolution (Borowiec et al., 2022; Christin et al., 2019; Han et al., 2023; Pichler & Hartig, 2023; Tabak et al., 2019). Although AI is not new, the increasing availability of resources such as GitHub Copilot, ChatGPT and Dall-E is leading to more researchers and students using these tools for research and study. While there is widespread concern within the community about the use of AI tools (van Dis et al., 2023), our focus in this perspective is on the positive opportunities for responsibly using AI, specifically large language models (LLMs), to assist with coding, both for research and in the classroom. We provide recommendations for publishing research that has used 'LLMs', which we take herein to include LLMs and comparable generative AI tools, but not more widely used aids such as spell-checkers, Grammarly and Writefull.

1.1 | What are LLMs and how do they work?

Artificial intelligence (AI) is the ability of a machine, generally a computer or robot, to perform tasks normally associated with intelligent humans (for an extended discussion of the merits of calling AI 'intelligent', which is well outside the bounds of this perspective; see Searle, 1997). Current discussion mostly centres on forms of generative AI, in which computers are able to produce content based on a set of training data. There are many types of generative AI, including several with which ecologists and evolutionary biologists will have some familiarity (e.g. machine-learning, deep learning, neural networks). Our focus here is on LLMs, which are machine learning algorithms that can perform various natural language processing tasks, such as classifying and generating text (including computer code), and responding to questions in a conversational style. There are several available LLMs, including ChatGPT (see Box 1), Gemini (previously Bard), Llama, Guanaco and OpenLLaMA, and many more are in development.

In simple terms, LLMs such as ChatGPT aim to predict the next part of a word or phrase based on a user-provided text prompt. The prediction is made by passing the prompt through a deep neural network, which itself has been trained to find relationships among words and phrases in an enormous corpus of training data (the 'large' in LLMs refers to the massive amount of data used to train these models). The predictions of LLMs are based on the content and structure of the corpus used to train them, including rules of grammar and syntax, how often words are found in a particular sequence and even how facts and ideas are presented together (regardless of whether or not that presentation is actually correct). Contextual clues also are used to improve the accuracy of the predictions, and models can be (imperfectly) trained to reduce the prevalence of certain problematic words and phrases (e.g. violent, racist or illegal ones). Rather than always selecting the absolute best option, most LLM algorithms build in a degree of randomness into their responses. This randomness has consequences for reproducibility (see below) though several tools now allow implementations where the user can remove this random component (e.g. ChatGPT v 4.0).

BOX 1 What is ChatGPT?

ChatGPT (https://chat.openai.com) was launched in late 2022 (Open AI, 2022), and is the most popular (and infamous) LLM currently available. It uses a generative pre-trained transformer (GPT) LLM that functions as a chatbot, allowing users to have interactive conversations to generate content. ChatGPT is popular because it is (still) free, although you need to pay for the newest version (at the time of writing version 3.5 is free, and version 4.0 must be paid for). It is a 'very large' LLM, which means it performs better than other 'large' LLMs. It is also relatively versatile, allowing users to apply it to multiple kinds of tasks. Note that although it is the most commonly used LLM application, ChatGPT is not always the best tool for what you want to do; other AI tools, such as GitHub Copilot and Llama, are optimised for programming. The LLM market is changing rapidly, so being flexible and investigating new tools as they emerge is likely to be a good strategy.

2 | USING LLMs FOR CODING

Large language models and other AI applications are uncannily good at generating well-functioning and understandable computer code based on natural language prompts from users—including users without a programming background (Ellis & Slade, 2023). Within ecology and evolution, many researchers and students already are using LLMs to help them write code (Duffy, 2024). Doing so requires two main skills: writing effective prompts and evaluating the responses. In addition, users need to know how to apply these tools responsibly.

2.1 | Writing effective prompts

Prompts are how you interact with an AI to generate a response. Prompts can include questions, comments, code snippets or examples, and should be written in full sentences in 'natural' language (i.e. plain text, as you might use to communicate with a colleague), not as a string of keywords as you would enter into a search engine. For example, if you wanted to learn how to run a linear regression in R, you might use the keywords 'linear regression' and 'R' in a search engine, but for an LLM, a better prompt would be something like 'Please show me how to perform a linear regression in R' (it is up to the user whether polite terms like 'please' are necessary). A good prompt will be detailed and specify as much context as possible (the who, what, when, where, why and how of the question).

For programming questions, prompts should include the programming language, any specific packages you want to use, and what you want to achieve with the code. Prompts also can be 'chained'; you can query the LLM with an initial prompt and then follow up with another prompt to update the response. Most LLMs intended for use by the general public will 'remember' the context provided in previous prompts within a single chat session (or multiple previous sessions depending on the tool in question), and use this to generate its responses. For example, if you ask for code to create a boxplot, in the next prompt you could ask it to change the colours, without needing to respecify the details of the boxplot. Another useful tip is to ask the LLM to 'explain with examples', or 'explain as if I am a high school student'. Note, however, that most LLMs have a character or token limit for questions. In ChatGPT version 3.5 (which is currently free), the limit is 4096 tokens or approximately 3000 words, which may limit the amount of detail you can put into your prompt.

2.2 | Evaluating the outputs

The hardest, yet most important, part of using LLMs to generate code is evaluating the accuracy of the code produced (Lubiana et al., 2023). An often underappreciated problem with AIs is that they 'hallucinate', that is, they can be completely confident in a response even when that response is inaccurate or entirely incorrect. Hallucinations tend to occur either when the prompts have not given the AI enough context to answer correctly, when the training data do not include sufficient information to address the prompt, or if the training data include mistakes. Hallucinations are more likely to occur when LLMs respond to prompts about less frequently used analytical methods, packages or programming languages. Because most LLMs are trained to produce answers that are perceived as correct by human editors, they also may favour grammatical correctness and plausibility over accuracy. It is therefore extremely important to be sceptical of any response given by any AI, and to always check that any code produced works in the way you want or expect it to. Consequently, to effectively use LLMs for coding, you still need to understand enough about the programming language to understand whether the outputs are correct or not.

2.3 | Other uses of LLMs in coding

In addition to generating code, LLMs can also assist with the following routine coding activities (see also Lubiana et al., 2023).

1. *Explain code (to yourself and others)*: It is not uncommon to have code or code snippets that you did not write (e.g. imported or modified from a paper, a collaborator, GitHub or Stack Overflow) or code that you wrote in the past and did not document well. With good prompts, LLMs can explain what the code is doing and why. A major advantage is that AIs are infinitely patient, so you can continue asking the same question repeatedly if the first explanation does not make sense.
2. *Commenting code*: Commenting is key to good, reproducible code (Cooper & Hsing, 2017), but some routine commenting is often skipped. LLMs can quickly comment code, including routine sections, saving researcher time and effort, which can be applied to commenting on more complex or bespoke sections of the code. Again, verifying the generated comments is critical.
3. *Translate code*: Many researchers are familiar with only one programming language, but may find they need to use another language to solve certain problems. LLMs can translate code from one language to another (e.g. from Python to R) or from one package to another (e.g. from tidyverse to base R).
4. *Debug code*: If your code is broken, you can provide the code to an LLM and ask it to find any errors. Ideally you should include any error messages and the aim of the code as context to your question. LLMs can be particularly good at explaining arcane error messages.
5. *Optimise code*: Sometimes it is easier to quickly write code that works, rather than spending a lot of time optimising the code to make it efficient and fast to run. LLMs can take unoptimised code and edit it to make it run faster and more efficiently. However, the correct metrics of optimization are important to verify beforehand.
6. *Unit tests*: Standard unit tests for functions are critical for many algorithms and packages. LLMs can assist in suggesting which tests would be most suitable and how to structure and write them.

You must check that the AI has done these routine coding activities correctly. In particular, debugging with LLMs can be rife with errors—recall that most LLMs are trained to produce results that look plausible to a human, meaning that errors in the code can be difficult to find.

2.4 | Benefits and challenges of using LLMs for coding

The primary benefit of using LLMs to generate code is that it is often faster than writing it ourselves, especially if we are non-professional programmers (i.e. most of us in ecology and evolution) or for tasks and problems that we have not encountered before. LLMs can lower the opportunity cost of trying something that might be complex to learn independently, and can speed up routine tasks leaving more time for other things like synthesis and idea development. LLMs also increase equity and inclusion, as they provide more opportunities for people without coding skills, with neurodivergent traits, who have less fluent English skills, and others who may benefit from different ways of working (Box 2).

Furthermore, LLMs can generate more than one suggestion for running a given data analysis task, complete with implemented and well-commented solutions and code examples. This not only can increase the rate at which coding skills can be learned but also may enable less-seasoned scholars, particularly students, to rapidly develop complex, multi-step methods for analysing a specific dataset, compare the outputs and choose the most appropriate way(s) to interpret the results.

The challenges of using LLMs for coding include the fact that responses vary greatly in quality—often in ways that are not readily apparent. For commonly used, well-documented functions, packages and languages, LLMs are typically fast, efficient and largely accurate because they have more examples of these in their training datasets; for example, answers using R packages like ggplot2 tend to be correct. Hallucinations are more common, however, when asking about less-used packages; for example, fitting phylogenetic comparative models using OUwie (Beaulieu & O'Meara, 2022) often give nonsensical results. Many LLMs also struggle with mathematics and logic, and need efficient prompts and extensive testing to debug or identify the errors (Chang et al., 2024; López Espejel et al., 2023). Responses also are typically not reproducible (at least for regular users), so different people may get different results with the same prompt. This inconsistency can be disconcerting to novice users.

BOX 2 Equity and inclusion with LLMs.

Debate around LLMs and inclusion tends to focus on the biases inherent in these models (Schwartz et al., 2022). If the training data being used to generate responses are biased (and we know that it is), then LLMs naturally will reflect this bias in their outputs, including in code outputs. This is obviously undesirable. Many developers of LLMs are trying to fix this issue, but training data are still focused on materials that are available in languages that are well-represented on the internet (both computer and human), and that are produced in wealthy nations of the Global North. Although there are arguably fewer cultural and regional differences in code than in say, prose or art, some of the same concerns are likely to apply with regard to the audiences for whom the tools are produced, the goals that they are meant to further, and the way that they are applied. New LLMs are also expensive to develop, in terms of staff, equipment and infrastructure, and their development will, in all likelihood, continue to be dominated by those with the most money and power. An additional large concern is that currently free platforms may either cease to be free, or stop being supported and developed; either way, the result will be to widen the existing gaps in computing and programming resources among institutions and world regions. It is important to ask who is benefitting from AI and who is missing out.

There are also potential equity and inclusion benefits of LLMs, especially for coding. The availability of free AI tools should help to increase opportunities for those who do not have coding skills, access to coding education or the ability to pay to learn them. Already, LLMs are being used in classrooms to provide bespoke advice and feedback to students. A student may be more comfortable asking a chatbot for help than a teaching assistant or lecturer, especially if they need to ask the same question multiple times to understand the answer, or students may be able to use LLMs to get those answers in a language or style that is more approachable to them. Teaching students to use LLMs for coding also can reduce inequities introduced by some students finding coding much harder than others. There are special advantages for neurodivergent students and students whose first language is not English. These benefits should also be considered when thinking about equity and inclusion in AI.

3 | LLMs AND CODING IN THE CLASSROOM

As long as LLMs remain easy to access (and particularly while they remain free to use), students will use them whether educators like it or not. Our recommendation is to guide and advise students on responsible use of AI rather than attempting to regulate or ban its use (see also Duffy, 2024; Lubiana et al., 2023; and https://cs50.ai/). Our own experience is that LLMs can be excellent aids for teaching and learning coding. Students can use LLMs to generate, explain, comment, translate, debug and optimise code. Students also can use AIs as personal tutors, and continue asking the same question

repeatedly in different ways as they work to fully understand the answers. Students also may be happier to ask chatbots for help than to ask a human instructor, out of fear that the latter will judge them harshly for not knowing the answers to simple questions. Likewise, LLMs can be a real help for instructors in crowded classrooms by providing a first point of contact for student questions. This facility may be particularly valuable for neurodivergent students or students whose first language is not English (or whatever the working language of the class in question is) and naturally shy students (e.g. Liu et al., 2024; Box 2). Two especially helpful features are that most LLMs can readily provide answers in multiple languages and they can quickly summarise or translate text from user manuals or help forums.

3.1 | Should we still teach or learn coding?

One commonly asked question when discussing LLMs and coding is whether we still need to teach coding skills or learn them as researchers, or if we should just outsource the process to AIs. We think that we should still teach students to code. Although LLMs can facilitate learning by students who otherwise struggle with coding, interpreting the outputs of LLMs and determining whether they are accurate still require a basic understanding of coding, statistics and mathematics. In addition, we must be able to detect when and why the code generated by an LLM is not working. Students also need to know what to ask the LLM to do, and why. Thus, coding skills will continue to be vital for students in ecology and evolution, in the same way that learning the basics of arithmetic is necessary for the effective use of a calculator or spreadsheet. We anticipate that simple debugging features and repetitive tasks can be identified and addressed easily with LLMs, saving time and efficiency, just as spelling and grammar checkers speed up proof reading of manuscripts.

Teaching coding skills also provides pedagogical benefits. For instance, learning to code requires us to break down large problems into smaller chunks that are usually easier to solve. It also requires strong logic and scientific reasoning to understand what you need to do and how to get a machine to do it for you. Moreover, developing these skills is a strategic process that can help to maximise creativity and create novel material, both of which are essential for doing science, making discoveries and spurring innovations (e.g. Fletcher & Benveniste, 2022). In summary, critical thinking skills will continue to be important, even in a world replete with text, video and code generated by LLMs.

4 | BEST PRACTICE FOR PUBLISHING CODE GENERATED USING LLMs

Current journal policies for publishing code at *Methods in Ecology and Evolution* (*MEE*) require that code needs to be novel, usable and understandable. *MEE* places emphasis on the quality, usability, accessibility and functionality of code (see https://besjournals.onlinelibrary.wiley.com/hub/editorial-policies). Concern has been raised about the accountability and transparency of publishing code generated using AI (e.g. van Dis et al., 2023). Who is accountable for the code (or text) and who should get credit for it? Most journals, including *MEE*, expect that the corresponding and senior authors of a paper are accountable for all of its contents. All authors are asked to read and agree to the submission of a final draft of a manuscript before submission. We know that not every author reads all the code associated with a paper, but at least one author must do so, and take responsibility for it. Using an LLM to help generate the code does not change this fact. When using an LLM, we would expect the responsible author(s) to follow the same kinds of quality assurance checks as they would if the code was written by them independently.

Going forward, when LLMs are used to generate code in publications at *MEE*, we will require the following:

1. At least one author must take responsibility for all associated code (or text) generated by the LLM. This responsibility must be explicitly noted in the Author Contributions section.
2. The use of AI/LLMs must be clearly stated in the manuscript in the Methods section. The AI application (e.g. ChatGPT) and version (e.g. 3.5) must be reported, along with details of how much of the content was generated by the AI.
3. The portions of code generated by the LLM must be annotated with comments stating that they were generated in this way.

Note that our focus here is on using LLMs for coding; the British Ecological Society (BES) journals, including *MEE*, also have a policy on AI-generated content more generally (https://besjournals.onlinelibrary.wiley.com/hub/editorial-policies), which also must be followed. One important part of this is that an AI cannot be considered as an author.

Another critical concern regarding code produced by LLMs is a lack of reproducibility. It is true that if you ask most LLMs the same question repeatedly, they will give different, and perhaps inaccurate, answers—but how much of a problem is this? Two researchers working on the same problem may produce different code solutions; indeed, one researcher working on the same problem at different times may do so, too. There will be specific situations where this will be a problem (e.g. using an LLM to collate data for a meta-analysis), but for basic code generation, this lack of reproducibility is not an issue as long as the code itself generates reproducible results and meets MEE's criteria of quality, usability, accessibility and functionality (https://besjournals.onlinelibrary.wiley.com/hub/editorial-policies), and that the authors acknowledge the use of LLMs in generating the code. To this end, it appears to us that the lack of reproducibility related to LLMs is similar to that of other tools—the results should be replicable but the precise process that went into deciding how to create those specific results may not be. In our opinion, that is simply how the creative process of science works. Being transparent in the use of AI and LLMs will continue to increase accountability.

More generally, we also caution that any uses of LLMs and AI require careful consideration of credit and liability. These considerations are relevant in terms of both intellectual contribution (i.e. who came up with the relevant ideas) and intellectual property (i.e. who has copyrighted the material that you or your LLM are using). Both the United States and the European Union are in the midst of enacting regulations on the topic, and, in most cases, it appears that if an LLM draws heavily on copyrighted text or images in the production of a published output, the human author may be held liable for copyright infringement. Similarly, if an LLM reproduces illegal contents (e.g. hate speech or symbols from banned political parties) that are then included in a published work, then the human author, not the LLM, likely will be held responsible for its products.

5 | CONCLUSIONS

AI and LLMs are not new technologies; various AI applications have existed since the 1960s. Many researchers concerned about the rise of LLM applications like ChatGPT forget that spell checkers and autocomplete functions, which we use every day in many different settings, also are various types of AI. However, the capabilities of generative AI and LLMs are increasing rapidly, and new developments appear almost daily. It is partially this speed of change, and the feeling of not being able to keep up, that is driving concerns. For example, that the AI in Excel reformats genetic sequences as dates is, after decades of pain, relatively common knowledge, but what do we know about similar bugs in tools like ChatGPT or Gemini? Change does not always have to be bad, however, and AI and LLMs are not going away anytime soon. We need to engage with these tools proactively and responsibly while promoting the best available practices, which themselves will continue to change. We accept that AI is a rapidly evolving field and we expect that our thoughts about it also will continue to evolve. We also note that we have different levels of expertise in using and developing AI tools; even experts do not know where the field will be a year from now. However, we still think—at least for now—that the potential positives of using LLMs for coding outweigh the potential negatives.

We also recognise the challenges inherent in these methods. Training LLMs has a staggeringly large environmental impact (Box 3), and these tools have the potential to increase global inequities (Box 2). As a community, we should consider how to use AI tools most effectively and ethically, how to improve transparency, who is benefiting, who is missing out and how we can reduce their environmental impacts. Ultimately, we think that AI is not here to replace us, but rather to assist us (more like the robots in Asimov's *I Robot* books than those in Čapek's *R.U.R.* or *The Terminator* films). However, given the rapid advances in AI, it probably would not hurt to add 'please' and 'thank you' to your ChatGPT prompts. Just in case.

BOX 3 The environmental impact of LLMs.

There has been much public concern about the effects of AI on academic integrity but there has been far less discussion about the direct and indirect environmental impacts of AI (Jay et al., 2024; Rillig et al., 2023; van Wynsberghe, 2021). CO_2 emissions for model training and tuning for just one natural language processing model have been estimated to exceed the average lifetime CO_2 emissions for a person living in the USA (Strubell et al., 2019). Using ChatGPT-like services in a single year produced 25 times the carbon emissions of training GPT-3 (Chien et al., 2023). LLMs also require a lot of infrastructure and equipment, all of which have associated environmental impacts; examples include water use and contamination, mining for rare-earth elements and the energy required for temperature control of servers (Rillig et al., 2023). Much current research is focussed on creating sustainable, lower-carbon LLMs (Chien et al., 2023; Patterson et al., 2021), but until these are successful, it is worth being very circumspect about the unnecessary use of AI for simple tasks. You might be adding substantially to your carbon footprint just to save a couple of minutes of effort.

AUTHOR CONTRIBUTIONS

Natalie Cooper wrote the first draft with input from other authors. All authors edited the manuscript and approved the final version for submission.

ACKNOWLEDGEMENTS

Thanks to the attendees of our 'Coding with ChatGPT and other LLMs' workshops at the BES 2023 Annual Meeting in Belfast, Natural History Museum London, and University of Sheffield, and to various contributors on social media, for discussion and suggestions which informed this perspective. ChatGPT v. 3.5 wrote the title under supervision of NC. The title was gently rearranged for clarity by AME.

FUNDING INFORMATION

NL was supported by NSERC and the Canada Research Chair program.

CONFLICT OF INTEREST STATEMENT

We are all editors at British Ecological Society (BES) journals, and (excluding ATC) we are compensated by BES for our work, thus we have vested interest in the adoption of these guidelines.

PEER REVIEW

The peer review history for this article is available at https://www.webofscience.com/api/gateway/wos/peer-review/10.1111/2041-210X.14325.

DATA AVAILABILITY STATEMENT

This paper uses no new data.

ORCID

Natalie Cooper https://orcid.org/0000-0003-4919-8655
Adam T. Clark https://orcid.org/0000-0002-8843-3278
Nicolas Lecomte https://orcid.org/0000-0002-8473-5375
Huijie Qiao https://orcid.org/0000-0002-5345-6234
Aaron M. Ellison https://orcid.org/0000-0003-4151-6081

REFERENCES

Beaulieu, J. M., & O'Meara, B. (2022). *OUwie: Analysis of evolutionary rates in an OU framework*. R package version 2.10. https://github.com/thej022214/OUwie

Borowiec, M. L., Dikow, R. B., Frandsen, P. B., McKeeken, A., Valentini, G., & White, A. E. (2022). Deep learning as a tool for ecology and evolution. *Methods in Ecology and Evolution*, *13*, 1640–1660. https://doi.org/10.1111/2041-210X.13901

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, *15*, 1–45. https://doi.org/10.1145/3641289

Chien, A. A., Lin, L., Nguyen, H., Rao, V., Sharma, T., & Wijayawardana, R. (2023). Reducing the carbon impact of generative AI inference (today and in 2035). In *Proceedings of the 2nd workshop on sustainable computer systems* (pp. 1–7). Association for Computing Machinery. https://dl.acm.org/doi/10.1145/3604930.3605705

Christin, S., Hervet, É., & Lecomte, N. (2019). Applications for deep learning in ecology. *Methods in Ecology and Evolution*, *10*, 1632–1644. https://doi.org/10.1111/2041-210X.13256

Cooper, N., & Hsing, P.-Y. (2017). *A guide to reproducible code in ecology and evolution*. BES Guides to Better Science. https://www.britishecologicalsociety.org/wp-content/uploads/2017/12/guide-to-reproducible-code.pdf

Duffy, M. (2024). Generative AI and graduate training in ecology. Dynamic Ecology Blog. https://dynamicecology.wordpress.com/2024/01/15/generative-ai-graduate-training-in-ecology/#more-66157

Ellis, A. R., & Slade, E. (2023). A new era of learning: Considerations for ChatGPT as a tool to enhance statistics and data science education. *Journal of Statistics and Data Science Education*, *31*, 128–133. https://doi.org/10.1080/26939169.2023.2223609

Fletcher, A., & Benveniste, M. (2022). A new method for training creativity: Narrative as an alternative to divergent thinking. *Annals of the New York Academy of Sciences*, *1512*, 29–45. https://doi.org/10.1111/nyas.14763

Han, B. A., Varshney, K. R., LaDeau, S., Subramaniam, A., Weathers, K. C., & Zwart, J. (2023). A synergistic future for AI and ecology. *Proceedings of the National Academy of Sciences of the United States of America*, *120*, e2220283120. https://doi.org/10.1073/pnas.2220283120

Jay, C., Yu, Y., Crawford, I., Archer-Nicholls, S., James, P., Gledson, A., Shaddick, G., Haines, R., Lannelongue, L., Lines, E., Hosking, S., & Topping, D. (2024). Prioritize environmental sustainability in use of AI and data science methods. *Nature Geoscience*, *17*, 106–108. https://doi.org/10.1038/s41561-023-01369-y

Liu, R., Zenke, C., Liu, C., Holves, A., Thornton, P., & Malan, D. J. (2024). Teaching CS50 with AI: Leveraging generative artificial intelligence in computer science education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science education V. 1 (SIGCSE 2024, March 20–23, 2024)* (pp. 1–7). ACM. https://doi.org/10.1145/3626252.3630938

López Espejel, J., Ettifouri, E. H., Yahaya Alassan, M. S., Chouham, E. M., & Dahhane, W. (2023). GPT-3.5, GPT-4, or BARD? Evaluating LLMs reasoning ability in zero-shot setting and performance boosting through prompts. *Natural Language Processing Journal*, *5*, 100032. https://doi.org/10.1016/j.nlp.2023.100032

Lubiana, T., Lopes, R., Medeiros, P., Silva, J. C., Goncalves, A. N. A., Maracaja-Coutinho, V., & Nakaya, H. I. (2023). Ten quick tips for harnessing the power of ChatGPT in computational biology. *PLoS Computational Biology*, *19*, e1011319. https://doi.org/10.1371/journal.pcbi.1011319

Open AI. (2022). ChatGPT: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L. M., Rothchild, D., So, D., Texier, M., & Dean, J. (2021). Carbon emissions and large neural network training. *arXiv*:2104.10350. https://doi.org/10.48550/arXiv.2104.10350

Pichler, M., & Hartig, F. (2023). Machine learning and deep learning—A review for ecologists. *Methods in Ecology and Evolution*, *14*, 994–1016. https://doi.org/10.1111/2041-210X.14061

Rillig, M. C., Ågerstrand, M., Bi, M., Gould, K. A., & Sauerland, U. (2023). Risks and benefits of large language models for the environment. *Environmental Science and Technology*, *57*, 3464–3466. https://doi.org/10.1021/acs.est.3c01106

Schwartz, R., Vassilev, A., Greene, K., Perine, L., Burt, A., & Hall, P. (2022). *Towards a standard for identifying and managing bias in artificial intelligence*. National Institute of Standards Special Publication, 1270. https://doi.org/10.6028/NIST.SP.1270

Searle, J. R. (1997). *The mystery of consciousness*. The New York Review of Books.

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 3645–3650). Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1355

Tabak, M. A., Norouzzadeh, M. S., Wolfson, D. W., Sweeney, S. J., Vercauteren, K. C., Snow, N. P., Halseth, J. M., Di Salvo, P. A., Lewis, J. S., White, M. D., Teton, B., Beasley, J. C., Schlichting, P. E., Boughton, R. K., Wight, B., Newkirk, E. S., Ivan, J. S., Odell, E. A., Brook, R. K., … Miller, R. S. (2019). Machine learning to classify animal species in camera trap images: Applications in ecology. *Methods in Ecology and Evolution*, *10*, 585–590. https://doi.org/10.1111/2041-210X.13120

van Dis, E. A. M., Bollen, J., Zuidema, W., van Rooij, R., & Bockting, C. L. (2023). ChatGPT: Five priorities for research. *Nature*, *614*, 224–226. https://doi.org/10.1038/d41586-023-00288-7

van Wynsberghe, A. (2021). Sustainable AI: AI for sustainability and the sustainability of AI. *AI and Ethics*, *1*, 213–218. https://doi.org/10.1007/s43681-021-00043-6

How to cite this article: Cooper, N., Clark, A. T., Lecomte, N., Qiao, H., & Ellison, A. M. (2024). Harnessing large language models for coding, teaching and inclusion to empower research in ecology and evolution. *Methods in Ecology and Evolution*, *15*, 1757–1763. https://doi.org/10.1111/2041-210X.14325