Reccommender:
Thank you for your wonderful “How to” article! It is a useful and concise read that should be helpful for many researchers. Two reviewers who have expertise in code sharing and/or promoting open research practices have provided very positive feedback and some helpful ideas that you might find useful to incorporate.
We thank the recommender and the two reviewers for the useful comments and suggestions.

If you decide to incorporate the addition of co-authorship for code reviewers, as suggested by Reviewer 1, please also reference a guideline for authorship to ensure that researchers are aware of what the code reviewers would need to do to fully earn authorship. For example, according to the ICMJE guidelines (http://www.icmje.org/recommendations/browse/roles-and-responsibilities/defining-the-role-of-authors-and-contributors.html#two), authors need to have contributed to the development of the article AND the writing of the article. Therefore, a code reviewer could earn authorship if they review the code (contributing to the development of the article) AND they help with the editing of the article.
Whilst we appreciate the in depth and insightful comments made by R1, we have decided to not add their suggestion of co-authorship for code reviewers during peer review as implementing such a shift in journal policy is far beyond the scope of this paper (although we appreciate the suggestion and more discussion is indeed needed).

If you decide to incorporate the discussion around offering reviewers co-authorship, as suggested by Reviewer 1, please also provide ideas for how peer review processes can address the issue of it being very difficult to find enough reviewers in the first place. If the few people who accept reviews were to become co-authors because of their code reviewing work as part of the review process, then new reviewers would need to be recruited to be the reviewers of the article (because authors cannot review their own articles).
As we aren’t going to include the mention/idea of reviewers becoming coauthors, for reasons given above, we won’t be adding this suggestion.

I have only a few minor comments:
- Line 109: perhaps change “and mistaking the column order” to “and producing a mistaken column order”
Changed.
“This type of error could easily be conceptual, arising from a misunderstanding of the dataset, or programmatic, such as from indexing by number and producing a mistaken column order.”

- Line 113: by “number” in “These errors are thought to scale with the number and complexity of code”, do you mean the number of lines? Or the number of code chunks? Or something else?
We agree this was vague and have made this more obvious in text.
“In particular, these errors are thought to scale with the number of lines and complexity of code (Lipow, 1982).”

- Line 116: wow, I had no idea about identical() - what a useful tool!
We’re glad you found this useful.

- Figure 2: it’s nice that you suggest contacting the authors directly. This can save so much time in the peer review process and promotes collegial interactions
Whilst we think that collegial interactions are important - it should be noted that this is intended for post-publication. This might not be possible during peer review due to

double-blinding, although most of the workflow remains the same in terms of checking through code to see if code matches methods, if it runs, if it reproduces etc.

- Line 183: for some reason the URL https://github.com/pditommaso/awesome-pipeline is not working - the pdf seems to be cutting it off, which results in a 404 error
I think this was an error with formatting, it should now work.

- Line 209: is Dryad free? I thought it cost money for authors to use it (which might be hidden by contracts Dryad has with publishers or universities).
Good point! We have removed mention of Dryad as it is not free.
"(e.g., stored on a free data repository such as The Open Science Framework (OSF), Zenodo, or for ecology data, the Knowledge Network for Biocomplexity)..."

- Figure 3: "Can my code be understood?" perhaps change to "Is my code understandable?". I'm not sure what a style guide is - maybe it is in the resources you suggested for cleaning up code? Regardless, make it a bit more obvious what this piece is
We have updated this part of the figure along the suggested lines.

- Line 276: this link is broken https://github.com/SORTEE/peer-277 code-review/issues/8
We think it was a formatting issue, which has now been fixed.

- Line 295: "not to get bogged down modifying or homogenising style" I would add "by" as in "bogged down by modifying"
Changed.
"It is important for code review not to get bogged down by modifying or homogenising style; as long as code is readable, then coding diversity should be encouraged."

- Line 338: "These benefits are substantial and could ultimately contribute to the adoption of code review during the publication process." Adoption by whom? Journals?
Yes, we have added "by journals".
"These benefits are substantial and could ultimately contribute to the adoption of code review during the publication process by journals".

A couple of things I've learned from my own open workflows that you might find useful for the article (of course, don't feel pressured to include these just because I mentioned them):
1) THE easiest way I find to make my code runable by anyone anywhere is when I upload the data sheet to GitHub and reference it in the R code so it will easily run from anyone's computer (see an example of the code here: https://github.com/corinalogan/grackles/blob/6c8930fcd66105b580809ef761d63b9cff0cbd83/Files/Preregistrations/g_flexmanip.Rmd#L233)
2) Line 209: consider adding the following data repository to your list: Knowledge Network for Biocomplexity (https://knb.ecoinformatics.org/). It is free, University-owned, and for ecology data, as well as being easily searchable because their metadata requirements are extensive (thus removing the need for researchers to remember all of the metadata they should be adding).
These are both great points! We have added mention of this data repository in our text. We haven't added the GitHub link as this could be problematic if the individual was running the code without access to the internet.
"(e.g., stored on a free data repository such as The Open Science Framework (OSF), Zenodo, or for ecology data, the Knowledge Network for Biocomplexity)..."

-

Reviewer 1:

This commentary highlighted the lack of reproducibility as a long-lasting, systematical issue in ecology and evolutionary biology, where results heavily depend on statistical modeling and numerical simulations. The authors suggested a comprehensive guideline to include code review in the pre-submission, peer review, and post-publication process to ensure the validity and robustness of scientific conclusions. Together with other pioneers who advocate for reproducible research in psychology and computer science, the authors proposed a valuable and practical framework (i.e., 4Rs, flowchart for peer reviewers).
As an advocate for reproducible research myself, I agree entirely with the necessity and urgency to change the status quo in the publication process, which emphasizes and incentivizes too much on the scientific novelties, but comparably too little on the reproducibility. Despite being out of the scope of this paper, in an ideal world, scientific discoveries should be published only after stringent fact-checking and careful examination of the method. Before sweeping through the entire field of biology, the discipline of ecology and evolution and other computationally extensive disciplines could be a pioneering test ground for incorporating code review as a standard publication practice.
In light of the well-versed rationales and the recommendations in this paper and my alignment with the preference for reproducible research, I do not have any major "concerns" overall. Before going to minor technical comments, I would like to share some general thoughts that may interest the authors to discuss or further develop in the revised manuscript.
We thank R1 for their insightful comments and suggestions.

1. In the section Are results reproducible?
Although each R in the 4R guidelines is indispensable, the intrinsic demands are increasing from the first to the last R. A fair and feasible implementation of these principles could vary by discipline/subdisciplines. Reproducible results have, of course, the uttermost importance and should be ensured whenever possible. Yet, in some cases, fully reproducing results is practical.
This is a good point and we already mention that the manner and scope of code review varies depending on the position in the research cycle on L66-69. We also mention that reviewers should be expected to at least check that code is reported on L359-362.

For example, evolutionary biology has relied on and/or gradually will rely more on insights obtained from high-throughput sequencing (comparative genomics, transcriptomics, and other omics). Due to the nature of its high-volume, sometimes high-dimensionality, computations involved are expansive not only in terms of computational time but also involves in the accessibility and availability of resources, including High-Performance Cluster (HPC), # of CPUs, memories, and storage. If reviewers were required to reproduce results in the peer review process, installation and configuration of a substantial fraction of bioinformatic software are not trivial, even if resources are provided/subsidized for the reviewer.
One possible solution to this (which may be outside the scope of this paper) is to watermark (e.g., using MD5 sum) all intermediate results during the computation and only examine the reproducibility of the "scaled-down" version of results.
This is important to note and we have added a mention of techniques involving long-computational time on L137-140 and L250-252 and large data on L212-215. In terms of MD5, I'm afraid I have limited knowledge as to what this requires other than providing a way of watermarking an original image. This is a useful suggestion but one that I'm not sure many people from the fields of Ecology and Evolution will understand fully.
"In some cases, reproducing analysis from models can take considerable time to complete, for instance when re-running complicated Bayesian models or other techniques involving long computational time."

"Some level of tolerance must therefore be applied especially when dealing with stochastic methods in which parameter estimates will change between subsequent runs or with techniques that are computationally demanding and slow."
"If sharing data is inappropriate to your study (for example when dealing with sensitive confidential data) or if data is so large it cannot be easily shared, then a user can provide a sample of simulated data or a primer so that the code can be checked and read (Quintana, 2020; Hennessy *et al.*, 2022)."

2. In the section Is the code Reliable

Besides the reasons mentioned by the authors, many errors in code could often appear in user-defined analysis or helper functions. Toy examples, if not unit tests in the engineering fields, should at least be included to test if those functions are doing what they are expected to do.

This is a useful point to mention - we have included mention of user-defined functions on L114-116 and the use of unit tests on L116-L120.

"This type of error could easily be conceptual, arising from a misunderstanding of the dataset, or programmatic, such as from indexing by number and producing a mistaken column order or from user-defined functions."
"Although some coding techniques, such as explicitly indexing by column name or by performing unit testing of any user-defined function (see Cooper, 2017; relevant packages include as {testthat}, Wickham, (2011) in R or {pytest} in Python, Okken, 2022), can help avoid many of these mistakes, this type of error is common and also extremely difficult to pick up by anyone without deep familiarity with the dataset and code."

3. On incentives for code-review club and reviewers

As most researchers are scientists (some may be statisticians) by training, requiring them to know and implement the best practices in coding could be too demanding, especially for the first author, who happens to be a bench scientist working on a highly independent project. Besides recommendations proposed by this paper (e.g., code review club, discussing up front potential authorship for code reviewers), I would go beyond and argue that code reviewers are entitled to authorship, especially if they contributed significantly in making code reviewable (Fig. 3)

On the other hand, given the current unfair workload that reviewers commonly face, reviewers should be offered the authorship opportunity as one of the incentives, especially if fact-checking is time-consuming. Although (again) may be out of the scope of this paper, this could be facilitated by journals that adopt a double-blind reviewing process. Authors' consent on granting anonymous reviewers should be asked before the manuscript is sent out to external reviewers, and editors must withhold the information from reviewers until the final acceptance decision on the manuscript to ensure the objectivity of reviewing process. To mitigate the possibility that authors are exploiting the reviewer's fact-checking work, authors should also agree that the reviewer will be entitled to authorships if they (1) find major discrepancies in the code that change the direction of the corresponding conclusions, (2) provides the evidence of their workflow (3) authors subsequently can reproduce the issues that reviewers postulated.

As the authors pointed out, designated "Data Editors" have become standard practices in some journals. Designated "Data Reviewers" may be on the horizon in the future. In such cases, authorship to data reviewers as incentives could be discussed, and it might be favorable not to grant authorship to data reviewers if they do not assess the manuscript's scientific novelty.

This is obviously a very tricky issue and is largely out of the scope of this paper - particularly in reference to providing authorship during peer review. We had meant to offer incentives to reviewers of code mostly before publication, outside of the peer review process (see L 326: "Code review, outside of paper submission and the formal

peer review process, can have a large impact on an individual's project, from error-checking, to validation of appropriate statistical analyses."). Although we agree that more needs to be done to clearly appreciate the hard work by reviewers of papers and of code during peer review, conceiving of such a large change in formal peer review is beyond the scope of this work. We have added a statement detailing the need for a workflow of coding review: on L333:
"For instance, a situation may arise where a code reviewer(s) finds a major coding error which, when fixed after highlighting and reproducing the issue to the author(s), alters the subsequent results and conclusions of the manuscript. Ultimately, incentives should be relative to the impact of the reviewer on the project."

Minor comments:
4. Consider switching the orders of Fig.2 and Fig. 3 and updating the in-text reference
Done

5. Line 183-184 https://github.com/pditommaso/awesome-184pipeline not found
6. Line 276-277 https://github.com/SORTEE/peer-277code-review/issues/8 not found
We believe this was a formatting issue and should be fixed now.

7. Line 352: add in-text reference "(Fig. 1)" after if code is indeed adhering to the R's listed above.
Added.

8. Line 358: missing full citation of Stodden 2011, Light et al., 2014
Added.

9. Fig 1: "Code must be error-free": the language is vague, and I suggest revising it to "Is the code doing what it is supposed to do?"
We have updated the language to: "Code runs and completes as intended".

Reviewer 2
The manuscript describes a process by which code-review in the field of ecology and evolution could take place. The authors suggest following the 4 R's (Reported, Run, Reliable, Reproducible).
Throughout the paper the authors provide some suggestions for reproducibility and effective managment of package versions. A tool not yet mentioned that my be useful is the R package "packrat" which stores packages specific to the version as they are installed, can reload them across different machines and can be shared across users. I recommend adding this type of package to the manuscript as this (version control/reproducibility) is what it was built for. It would be nice to see a packrat snapshot submitted along with code to fully reproduce an analyses.
Another tool which can greatly aid in reproduciblity is containers and/or the containerization of a research project. Here is a paper describing their use in data science https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008316
And more information on the reproducibility of containers here: https://carpentries-incubator.github.io/docker-introduction/reproduciblity/index.html
Beyond that I found the paper to be well written and well structured and feel it will be a helpful guide for the community. Congratulations to the authors and thank you for putting this together!

I would love to see journals take code review serieously and hire someone to code review alongside the work performed by the editors and reviewers. Perhaps some of the society journals should take the lead here?

We thank the reviewer for these suggestions and have made a mention of both the packrat package (although this has since been superseded by the renv package) and containerisation through Docker.
"Packages such as {renv} (Ushey, 2023; which replaces {packrat}, Ushey *et al.*, 2022), {groundhog} (Simonsohn & Gruson, 2023), or {poetry} (Eustace, 2023) and {pipenv} (Pipenv Maintainer Team, 2023) in Python can help with ensuring a reproducible environment and allow for specific loading of desired package versions. Another option is containerisation through the use of Docker (Boettiger, 2015). Detailed tutorials already exist which highlight the use of this reproducible method in far more detail than we will discuss here."