**Molecular Ecology Resources Reviews**

We thank all reviewers for thorough and very helpful comments. We hope we have addressed their concerns with our revisions. In addition to addressing all of the reviewers issues we have also updated slimr to be compatible with the latest version of SLiM (v4.0) which introduced a number of breaking changes and new features that slimr was not able to take advantage of without being updated. Note that slimr now has experimental support for the most advanced new feature in SLiM 4.0, that of multispecies simulations, but we have not had a chance to extensively test this functionality before the deadline for these revisions. Therefore, multispecies support is currently an undocumented feature that we are testing and hope to release officially very soon (for any reviewers who are interested, using multiple species in slimr is as simple as using a named argument in slim\_block, where the name of the first argument should be the name of the species that the code block is relevant to). Nevertheless, single species simulations (which were the only possible before v4.0) are fully supported, and so all examples from our original submission work fine using SLiM 4.0 or greater. We respond to individual reviewer comments below.

**Reviewer 1**

*Overall comments:*

*Overall, the paper is written in a clear, compelling, and straightforward manner, and it seems like a great*

*fit for Molecular Ecology Resources. Nonetheless, persistent confusion arises from the unclear and*

*inconsistent differentiation between two distinct classes of prefixed functions. The homepage for the*

*software is clean and clear, and the software installed easily on my machine (Linux laptop, Ubuntu-*

*based operating system), but I was unable to recreate and run some of the basic sample code displayed*

*in the paper. (This also holds for the supporting package, slimrmodels, which has an incomplete*

*homepage, and which I was able to install but was unable to use as demonstrated in the paper.) The*

*supplementary Rmarkdown tutorial is a great resource to help walk new users through the package’s*

*Value!*

Thank you very much for the compliments on the package and the paper. We are sincerely sorry for the issues you experienced with running the package. We have made every effort to make the experience of using the package smoother. We agree about the potential confusion about the function prefixes, and have followed your suggestions to make them clearer (more on this below).

*The authors have clearly put together and put forth a well thought-out piece of software which has the*

*potential to help lower the ‘cost of entry’ to SLiM’s high-powered but complicated and idiosyncratic*

*simulation-scripting framework – and thus the potential to help broaden the reach of simulation*

*modeling, a crucial component of the population genetics toolbox. I am genuinely excited about that, and*

*excited to see the future work that it facilitates! That said, I believe that two major modifications are*

*needed in order to make the software ready for robust public use and to make the paper accurate and*

*Complete:*

Again, thank you for the kind words about the concept of the package.

*1.) both slimr and slimrmodels should be fixed, and then installed and test-run on all common*

*operating systems, to ensure that the functions and functionality displayed in the paper mirror what is*

*accessible when a user installs and runs the software (with sincere apologies if it turns out that I have*

*done something wrong in my attempt to use the software!);*

We have improved the extent and breadth of our package testing. We have set up a github actions workflow to test the package on all platforms available to github actions. We have attempted to cover the failure cases you have uncovered in our github actions testing. We also clarify below that some of the failures were due to author errors related to versioning that have now been fixed, but should not have happened in the first place. We appreciate your patience and tolerance for these errors, and we have tried to ensure that this revised manuscript does not have the same issues. We have decided that slimrmodels is not ready for public release. It works in local testing, but is not polished, has no unit tests, and is not properly documented. We do think it will be valuable, but since technically it is a separate package from slimr, we feel it is not necessary to include it in this manuscript. When it is ready for release we will do so (along with a software note of its own). We apologize for a lack of attention to this. The reason slimrmodels did not work for you was because in the final rush of finishing and submitting this manuscript I failed to push the updates to slimrmodels that made it functional to github. Nevertheless even with this basic functionality the package is not ready for full release due to the issues we mentioned above.

*2.) a review and possible redesign of the double-prefix function-naming conventions used by the*

*package, followed by a full edit, to bring the paper into agreement with the final conventions and to make*

*the paper’s explanations of those conventions internally consistent.*

We have redesigned the prefix naming convention in line with your suggestion and believe the package has been improved as a result. Thank you.

*Below I provide bulleted lists of my major and minor suggestions, for the authors’ consideration and use.*

*Major suggestions:*

*I suspect this is because of active package development, but I was unable to run the code demonstrated*

*in Figure 1. Once I had defined an R variable called ‘script\_1’ then I was able to run it using the*

*`slim\_run` command. My printed output looked different than the authors’ displayed output, in that it*

*didn’t contain any information about the number of generations and/or time elapsed. I assume that that*

*is no big issue however, as it probably just stems from a change in the program’s default output.*

*However, I then tried to output the results to a genlight data structure, but failed to do so. First I tried*

*running the `slim\_output\_genlight`, only to find that that function no longer exists. (Indeed, there are no*

*functions prefixed with “slim\_output\_”, and there are a number of “slimr\_output\_”-prefixed functions, as*

*discussed in the section starting at line 222, but “slimr\_output\_genlight” is not among them.) Then I used*

*tab-completion on “slim\_” to peruse the current slimr functions and try to find the likely new name for that*

*former function; I found “slim\_extract\_genlight”, and that seemed like a good candidate, but running that*

*produced an error. I provide a screenshot of this series of issues below. It may be that this is intended*

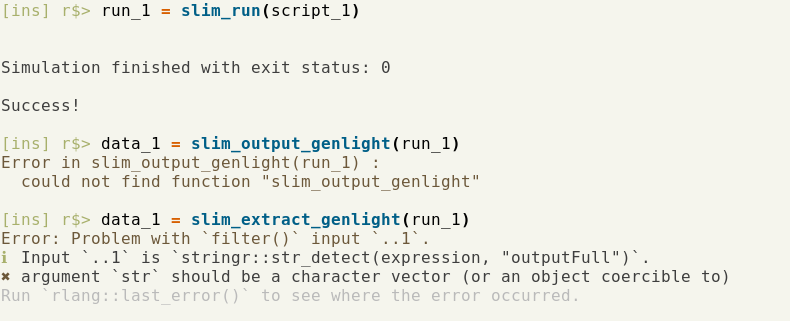
*behavior because the package has developed since this paper was written, but if this is the case it*

*seems like it might be best to either reproduce Figure 1 using current code, and/or indicate what version*

*of slimr was used to produce the results shown in Figure 1, to avoid confusing future readers. (Caveat: I*

*sincerely apologize if any of this is user error! If so, hopefully that is at least helpful for clarifying potential*

*gotchas for first-time users!)*



We apologize for the issues you had with figure 1. You indeed were correct in your suspicion that this was due to active development, where we had decided to rename the slim\_output\_ functions to slim\_extract\_. As to why using slim\_extract\_genlight did not work, it is because we changed what sort of object should be input into the function. This would not have happened if we had 'frozen' a version of the package to be used by the reviewers, and continued development on a new separate branch that would not change. We have done this for the revisions. We have also entered a new phase of development where we believe the main syntax of the package should be considered stable, and if changed in such a substantial way in the future, we will retain the old 'deprecated' version of any renamed function in the package for several releases before removing them (and accompanying the running of any such function with a warning to the user that they are using a deprecated function).

Figure 1 has been updated with updated syntax, and should work with the latest version of slimr now, but it no longer contains the code for extracting a genlight, instead it focuses solely on demonstrating autocomplete..

*The “slimr verb” idea is neat and clean, and makes a lot of sense. However, I found myself very*

*confused and struggling to grasp it after I read the initial portion of the “Data Input/Output” section. I*

*thought I easily understood the idea that “verbs have the prefix slimr\_, and are meant to be used only*

*inside slim\_script calls”, but “other slimr functions are prefixed with slim\_, which means they are to be*

*used on slimr\_script objects, and not inside a slim\_script call”. Then I looked at Figure 2, section A, and*

*was confused why both types of functions (“slim\_”-prefixed and “slimr\_”-prefixed) appear to be used*

*inside a single “slim\_script” call (e.g, “slimr\_block\_init\_minimal” and “slim\_block”). I may be dense and*

*completely misreading things, but if not then this should either be rectified and made consistent with the*

*text, or the text should be edited to reflect the mixed usage of the two types of functions. (Furthermore, if*

*this means that the difference between the two types of functions is not as clear-cut as the text implies*

*then I would suggest considering just simplifying everything to a single prefix, and/or reconfiguring the*

*two sets of functions.)*

Part of the issue you have uncovered is that when we said slimr\_ prefixed functions are meant to be used inside a slim\_script call, what we should have said was they were meant to be used inside a slim\_block call. slim\_block creates a block of SLiM code, so we consider it a SLiM development function. Anything inside slim\_block is meant to be SLiM code augmented with slimr verbs. But indeed this makes the separation between the two types of function less clear, so we have rethought the two categories as you suggest in multiple places and for multiple reasons. However, we are very happy that you like the idea of slimr verbs, we also think this is an elegant way to achieve their function. As for the issue in figure 2, indeed, slimr\_block\_init\_minimal() was a mistake. It should have been slim\_block\_init\_minimal(). We've fixed this in the latest version of the package. Though we don't mention it in the paper functions that starts with slim\_block\_ create slim\_block() code blocks. In this case it creates a minimal initialize code block from its arguments, instead of from typing out all the SLiM initialize functions.

*The “slimr\_output” and “slimr\_template” sections describe functions that appear to break the authors’*

*stated rule that “all slimr verbs have the prefix slimr\_, and are meant to be used only inside slim\_script*

*calls”. If so, this definitely needs to be corrected or disambiguated. (If not, my apologies for the*

*confusion! It is hard to keep track of the “slimr\_” vs. “slim\_” prefix difference.)*

Some misunderstanding seems to have happened here. slimr\_output and slimr\_template are meant to be used inside calls to slim\_script (or rather, slim\_block as we clarified above). They both modify the SLiM script in some way from R. We have rewritten these sections to try and make this more obvious. They have also been renamed (see our reply to a later comment for details on renaming).

*The first example is great! I just think it needs to be explained a more clearly. The sentence in 284-5*

*initially struck me as self-contradictory: If there is a probability of splitting in each generation, then I fail to*

*understand how the population could split into two populations “at some rate”, rather than*

*“instantaneously, at a randomly-drawn moment”. Then, after I read on further and reviewed Figures 3*

*and 4, I believe I realized what is actually being simulated: 1.) The authors are simulating a fixed-size,*

*discrete-generation meta-population consisting of N subpopulations (where N==1 at the start), and 2.) at*

*each time step (i.e., generation) a single subpopulation is chosen, a Bernoulli draw (with constant p) is*

*made, and that subpopulation is split into two new, equal-sized subpopulations if the draw == 1. If so,*

*this should be explained more clearly. I feel as though the statement “simulates a population of 100*

*individuals that randomly splits into two equally sized subpopulations at some rate” implies both that a.)*

*the population eventually has only 2 subpopulations, and b.) the split happens slowly (i.e., that the rate*

*of migration between the two subpopulations effectively decreases from m=1 to m=0 over some multi-*

*time-step time period) – but as I understand the results, neither of those things is true! Some clarification*

*could go a long way. (Caveat: I do not write Eidos/use SLiM, so I am not certain that my interpretation of*

*the code is correct, and if not, I apologize for the misunderstanding.)*

We have reworded. We meant the splits themselves happen at some rate (so in each time interval there was some number of splitting events, but probability is a better word than rate here). The splitting events themselves were instantaneous (so reproductive isolation occurs instantly with no protraction period). “instantaneously, at a randomly-drawn moment” is a more correct way of saying it. We have described the process in more detail.

*The idea of creating a separate R package to help share useful slimr simulations (slimrmodels) is great!*

*However, the landing page for the GitHub repository appears unfinished, and even though I was able to*

*successfully install the package it appears to me to have no contents, such that I was unable to run the*

*final code example provided in the paper. I suggest that this either be finished and fixed or else be*

*removed from the paper before publication.*

We are glad you like the idea, but apologise that the slimrmodels package was not quite as ready to go as it should have been. We have removed the section on slimrmodels so that we can give it the attention it deserves in a follow-up effort.

*This is certainly optional for the publication itself, but the package’s homepage could be built out a bit,*

*which might help ‘onboard’ new users. For example:*

*◦ Right now the homepage shows how to create a basic SLiM script and save it as an R variable,*

*which I could easily do. However, it then says that this could be exported to text to run in a*

*standalone SLiM installation, or run in SLiM directly from R (or many other things), and it points the*

*user to the slimr vignettes for this. However, when I run `vignette()` and browse or search my list of*

*vignettes, nothing from slimr appears. Were I not following along with the paper, this would leave me*

*in a lurch and frustrate me as a potential new user.*

Agreed. These vignette's were planned but then were not completed before we submitted the paper. They should have been better prioritized. These vignette's are now completed and accessible from the package website as well. Please look at the Articles menu to see a selection of vignettes.

*◦ The website could also be built out by hosting a few static walkthroughs of the available vignettes*

*and/or other example use cases (e.g., Rmarkdown scripts running and plotting the results of the*

*basic example simulations that are showcased in the paper). The simplest way I can see to achieve*

*this would be to just drop the outstanding supplementary HTML file into an ‘examples’ or ‘tutorials’*

*section of the GitHub repo, then provide a link to it within the landing-page README.*

Thank you for the suggestion. We have added links to several example vignettes to the main page.

*Minor suggestions:*

*I am not convinced on the utility of the two categories of “slim\_”- and “slimr\_”-prefixed functions. I believe*

*I understand the concept, and it feels important and helpful. I just find these particular prefixes to be*

*riddled with confusion because: a.) they are so similar that it is hard to note the difference while writing*

*code, leading to easy typos and frustration; b.) the “slim\_” prefix feels as though it should indicate*

*functions that are direct wrappers around SLiM commands, while the “slimr\_” prefix feels as though it*

*should indicate ‘higher-level’ functions that operate on SLiM-script abstractions that slimr has created,*

*but after reading the section from lines 191-203 it feels like it is closer to the opposite, if anything. I*

*understand it could be a headache, so would not blame you for rejecting this suggestion, but I would*

*nonetheless suggest exploring slightly longer but clearer and less confounding prefixes (e.g., perhaps*

*something like “slimr\_build\_” versus “slimr\_meta”, or simply “slimr\_build\_” versus “slimr\_”?).*

*There are a number of spots in the script where 'SLiM' is instead written 'SliM' (e.g., lines 89, 95, 114,*

*158, 162, 172, 173, ...). This is a very minor typo, but actually created some confusion at first because it*

*made me think that ‘SliM’ is not ‘SLiM’, but instead a separate piece of related software, much like the*

*software that the authors are introducing ('slimr'). Probably best to correct this across the manuscript, for*

*Clarity.*

We have revamped the prefix system and have made sure our spellings of SLiM are consistent. Specifically, we have retained the old prefixes so that code written using the old version of slimr still works (our own group for example has used it in a number of active projects and a few other labs are already starting to pick it up), but we have created new prefixed versions which are alias for the originals. We have settled on using the slim\_ prefix for functions that construct SLiM code and for functions that manipulate the resulting R objects that hold the SLiM code (and frankly all functions that are not slimr verbs). For the slimr verbs we have opted to use the very simple r\_ prefix. This signals that the functions are not SLiM functions but R functions (hence r\_), which are 'injecting' SLiM code using logic that is specified from within R, if that makes sense? It certainly makes the two prefixes easier to disambiguate. This means essentially all functions in slimr are prefixed slim\_ (to help with autocomplete and to reduce clashes with other package's function names), except for slimr verbs, which are only used in slim\_block calls (and which don't do anything if used outside that context).

*Line 114: To clarify, perhaps change to: “[...] the syntax used to write SLiM code in slimr is very similar to the native SLiM syntax [...]” (emphasis added, to highlight my initial confusion).*

Done

*Figure 1: Perhaps separate sections A, B, and C in a bit better. Initially it gives the appearance of a single, multi-paneled screen from an IDE like RStudio, which then feels confusing because it features 3 text editor panels and 1 running R CLI. Most simply, just increase the gray space between the separate subpanels. Alternatively, if the authors get rid of the gray background, separate the sections with bold black lines and/or bold black boxes, and then make the letter labels capitals, bold, and larger. It might also help to provide a bold arrow from the ‘before’ to the ‘after’ portion of the autocompletion in part A.*

Figure 1 has been regenerated to reflect changes in slimr and we have taken your suggestion with regards to the panel layout. It also now only focuses on demonstrating the autocomplete functionality.

*Line 177: I think it would be really helpful to indicate in parentheses precisely how “R users [can] look up relevant SliM functions in their R session”.*

Done

*Line 187: I believe this is intended to read: “can be run in SliM, and their results collected and returned, using the slim\_run function.”, or “can be run in SliM (and their results collected and returned) using the slim\_run function. Minor but helpful disambiguation.*

Done

*Lines 216-7: I would suggest clarifying (e.g., “This is a powerful way to use within a simulation empirical data that has been generated, loaded, and / or cleaned in R.”).*

Done

*Lines 226-7: I would clarify (e.g., “After calling `slim\_run` on a script object, the output will be available as a data.frame within the returned object.”).*

Done

*Line 236: Perhaps get rid of the second instance of “programming”?*

Done

*Line 258: Change “hybrid domain specific” to “hybrid, domain-specific”?*

Done

*Line 270: The function is alternately called “slim\_run” and “slimr\_run” in the same sentence. Choose.*

Done

*Line 275: “wil” to “will”*

Done

*Line 280: “the use slimr a short and simple” → “the use of slimr in a short and simple”*

Done

*Line 284: “a population 100” → “a population of 100”*

Done

*Lines 290-2: I would suggest rewriting this sentence for clarification. Perhaps something like, “The forcing operator forces R to replace the following statement with the value to which it evaluates, counter to R’s default form of evaluation (hence the term ‘forcing’).”*

Done

*Figure 3: Label the axes. And offer a bit more clarification in the caption (e.g., “subpopulation clustering is obvious as a pattern or strong horizontal bands.”). It could also be worth considering collapsing Figures 3 and 4 together, to align the subpopulations at the tips of the UPGMA tree with the rows of Figure 3 – but I understand that they may require more work and offer little marginal value.*

We considered your suggestion but decided to keep them separate so that we could keep the code examples simple (your suggestion is certainly possible but would require much more verbose code that we would be unable to explain concisely in the body of text).

*Lines 393-4: These lines appear to reference the figures associated with the previous section instead of the figures associated with this one.*

Indeed, this was leftover from a previous draft which we neglected to update. These figures have been removed in the updated manuscript in an effort to streamline the paper's presentation.

*Lines 482-3: Change “before after” to “before”.*

Done

*Line 509: “wrapped into R package” → “wrapped into an R package”*

Done

*Figure 6: Perhaps improve size and clarity of axis labels.*

Done

*Figure 6: May want to clarify that the subpopulation acronyms refer to the locations chosen and explained in the supplemental HTML file.*

Done

*Figures 6 and 7: I would suggest adding uncertainty bars to the 2006, 2007, and 2008 point estimates of FST from the field data, and a statistical test of the null hypothesis that FST is the same in all years, if possible. (I do not know off the top of my head what the best methods for this would be, but anything reasonable and justifiable based on methods applied in previous literature would lend some credence to the simulation setup, in my opinion.) The interannual variation appears quite small relative to a lot of the simulated FST values, and it appears as though it could easily arise from sampling error (suggesting a null hypothesis that FST is actually year-invariant, and suggesting a simple alternative explanation for the wide range of simulation parameter-value combinations that are found to generate relatively similar results). I understand that this ultimately does not matter from the point of view of a methods paper just aimed at demonstrating a piece of simulation software, but it would nonetheless help support the motivating concept. Otherwise, the setup fell a little flat for me, given that it felt as though the three years could have been chosen so as to appear to show something interesting worth simulating (despite the plausible and compelling mechanism posited to drive FST fluctuations over time).*

Thank you for the suggestion. This was added to the example vignette. We used simple bootstrapping to get confidence intervals and to test whether the years differed significantly (they did). We also clarified that the years were chosen to sequence based on rainfall data to test what effect the large rainfall events had on population genetics, they were not chosen based on their Fst, which was not known until after the samples had been sequenced.

**Reviewer 2**

*Comments to the Author*

*The authors present here an R package wrapping the forward in time population genomic software program SLiM, allowing to connect (and also extend) SLiM with the data manipulation and analysis capacities of R. Connecting these two platforms makes SLiM not only attractive to experienced R-users, but also allows to do everything within a single framework; from data preparation to data simulation until the data analysis.*

*General comments*

*The authors did a great work to combine SLiM with R. It is not just a(nother) wrapper, but integrates SLiM in R in a complete manner, overtaking all functionalities of the base SLiM program, including the help and the syntax highlighting. slimr extends the functionalities of SLiM allowing to run subsequent simulations without additional scripting in another language. I believe that slimr is a great tool and a great help for biologists.*

Thank you very much for the complimentary words. We are happy you think slimr will be a great tool; we tend to think so too (hopefully a little better now for the revisions).

*The manuscript is easy to read but gives me more the feeling of reading a manual than a paper. At the beginning the manuscript elaborates the implementation and its potential, then quickly goes into implementation details and examples before finishing with two more detailed simulation examples. Reading a manuscript, I am keen to learn what a tool can do and how superior it is to others. Some teasers (syntax highlighting for example) and potential analyses are also welcome. However, detailed examples I rather expect to find in supplement material than in a manuscript. The first example is still acceptable, but the second example is too exhaustive to be in the manuscript itself. Would be good if you could reduce the examples to describe the type of analyses one can do with slimr and to push the details to the supplement material.*

We have moved most of the second example to the Supplement, retaining only some of its results in the main manuscript.

*The implementation features sound great, but unfortunately, I had a hard time to get started and to run the example scripts. A tool where the installation process and then the examples do not work out of the box will have a hard time to be used. It can well be that my problems are based on the use of MacOS, but I believe that MacOS users are not as uncommon in population genetics that they should be ignored. Please see below for the problems I encountered.*

We certainly do not want to ignore MacOS users, as indeed, they are a substantial potential user base. However, none of the development team are MacOS users ourselves, making it hard to test and maintain the package for Mac users. We have set up automatic testing on MacOS through github actions, but these tests currently don't test the installation and setup of SLiM using slim\_setup(), but they do set up SLiM manually from the SLiM setup files.

Technical issues / minor comments

*That are the problem I encountered during the installation procedure of ‘slimr’:*

*- To use the R packages, I had to install the missing packages manually, which went fine for most of them. Problematic was only ‘dartR’. The installation of ‘dartR’ worked in principle fine, but when loading the package, it complained that ‘SNPrelate’ was missing which I had to install manually. Clearly it is not a slimr bug but would be great if it would work out of the box.*

*As you note, dartR is not a dependency of slimr, however it is used in the main example.*

In the main example we have added code for installing these dependencies, so that user's can run the example out of the box, even if they don't have these dependencies already.

*Running ‘slim\_setup()’ on my Mac did not work:*

*- First, by default MacOS has ‘wget’ not installed, so I had to install it manually*

*- Second, I think there was an issue with the unzipping, I am not anymore able to reproduce this problem as I gave up and installed SLiM manually using the installer provided by SLiM. This manual install went smoothly.*

We are glad your manual install went smoothly. slim\_setup() is admittedly an experimental function that will likely fail for many users because there are so many variations in system setup. It would be very difficult to make a function that would be robust to all these variations. We have added installation instructions for the more likely scenario of requiring manual SLiM installation, pointing users to the installation instructions for SLiM in the SLiM manual, and then explaining how to get slimr to find and link up to this local installation.

*What about adding a section for MacOS users how to get started. Similar to how it has been done for Windows users?*

As we previously mentioned, unfortunately none of our team uses MacOS and so creating a getting started guide for MacOS would be a major challenge. In theory, once installation instructions for SLiM have been followed, which are available in detail for MacOS, the remaining steps should be the same for all operating systems.We've tried to highlight this installation path because it likely will need to be followed by many users and is more challenging than if the automated setup works, so should be covered in more detail, as you suggest.

*Script ‘Simple\_example\_using\_migratio*

*n\_and\_fst.Rmd’ had a problem in line 176:*

*> plot(inds)*

*Processing genlight objectError in utils.check.datatype(x, verbose = verbose) :*

*Fatal Error -- SNP or SilicoDArT coding misspecified, run gl <- gl.compliance.check(gl)*

*Script ‘Testhat.R’ had a problem in line 4:*

*> test\_check("slimr")*

*Error in `test\_dir()`:*

*! No test files found*

*Run `rlang::last\_error()` to see where the error occurred.*

This script is only meant to be run internally by the testthat package to test that things are running okay. Running it outside this context will not necessarily work.

*Script ‘Custum\_vis.Rmd’ had a problem in line 151:*

*> sim\_results <- slim\_run(chromosome\_sim\_1,*

*+ callbacks = list(make\_hilbert),*

*+ cb\_args = list(max\_prev = 1000))*

*[=================================================>----] \ Gen:920/1000( 92%) Time: past:00:00:00|left: 0sError in assert\_ratio(ratio) : x <= 1 is not TRUE*

This vignette is fairly old but we have been able to run it recently without problem. If you still have the issue, perhaps it is platform specific. If so, we would appreciate it if you might post an issue to github (https://github.com/rdinnager/slimr/issues)

**Reviewer 3**

*Comments to the Author*

*In the presented manuscript, the authors introduce a new R package for developing and running SLiM simulations from the R environment.*

*From a technical standpoint, slimr represents a remarkable technical achievement. By utilizing clever metaprogramming techniques available in the R programming language, the authors provide a very interesting R interface to the widely popular population genetic simulation software SLiM, including a translation layer between SLiM constructs implemented in its programming language Eidos and standard R. In particular, the templating features could provide a very useful alternative for running SLiM scripts across parameter grids, which is something that’s normally done either by utilizing SLiM’s command-line parametrization, or by utilizing standard string templating available in R or Python. slimr is a great example of why R is such a powerful tool for data analysis due to its excellent support for building Domain Specific Languages.*

We appreciate these kind words about the technical merits of our package. Thank you!

*Unfortunately, despite the impressive amount of software engineering involved in developing such a tight integration between R and SLiM/Eidos, the manuscript itself does not do a good job in selling the R package to the audience. My main issue comes down to the presentation of code examples (many of which I had trouble reproducing) which hindered testing the features provided by the R package, experimenting with the examples and — ultimately — properly reviewing them.*

We apologize for the issues you had with the code examples in the manuscript and have endeavored to make sure they will be reproducible. Unfortunately, many of the issues you encountered were the result of continuing development on the package after submitting the manuscript. It was a major error on my part to not maintain a stable version for the purposes of your review. For the revision we have split off a branch of the slimr repository on github that will remain unchanged, so this sort of issue we hope does not happen again.

*I will write my comments for each example subheading separately.*

*# General comment on examples*

*Any paper which includes code (most certainly a paper describing a new programming library) should have those code examples easily available and executable directly from its documentation, without having to copy-paste them from a PDF, type them out from a screenshot, or copy them from a downloaded supplementary file. In case of R packages, these examples should be provided as a fully reproducible vignette distributed with the package and available on the website (such as the one slimr already has), including necessary data. There is no reason why this shouldn't be done.*

*In case of this paper, I had to either copy-paste code from the PDF to my R session (which often broke formatting in strange ways leading to syntax errors), some examples are shown as screenshots, where copy-pasting is impossible. One example is provided as a plain R script, another as a very long rendered HTML notebook. I would strongly suggest unifying all sources of code in this paper in a single, easily accessible form and fully reproducible as one or more vignettes (with code, results, and data(!) included) available on the package website.*

We agree and apologize for not including this in our previous submission. We have now added the examples from this manuscript as vignettes in the package and will also include the original .Rmd files in the submission for the reviewers use.

*If a reproducibility snapshot is needed for the journal, the authors can easily provide a tagged archive with the relevant git commit on Zenodo or simply state the git commit link for the snapshot in the paper.*

We have set up a release on github (and zenodo) for reviewers of the revisions (tagged v0.2.0). We have provided instructions for how to install the correct version as a supplementary document for the reviewer use only.

*Now on my comments on each individual example (one comment per paragraph):*

*## Example: "Integrated Development"*

*Fig. 1 (panel a) is probably intended to illustrate some autocomplete features of slimr. However, when I manually typed the `sim%.%.SS$` as shown in the image, I don’t get any autocomplete at all. Did I totally misunderstand what’s supposed to happen there?*

*Autocomplete of SLiM constructs in R is arguably one of the main selling points of slimr (without that, SLiMgui’s tight integration of code and documentation is much more powerful, despite the convenience trade-off of having to write Eidos and not R). This feature needs to be explained in the main text in detail. For instance, what is the `sim` object in `sim%.%.SS$…` (it does not appear to be exported in the slimr namespace)? Where does it come from? An OOP-style "dot-notation" is not applicable in R, so standard autocomplete on `sim.<tab>` will do nothing (unlike, say, Python). How is the autocompletion supposed to work here (for instance, how can the user autocomplete something like `sim.simula<tionFinished()>;` on line 20 in Fig. 1a? What is this `sim%.%.SS$` supposed to be?*

Apologies for this. Again, this was the result of ongoing development, where we decided to change the names of the objects representing SLiM classes. .SS is now simply SS (and also can be accessed as SLiMSim).

*Regardless of this technical problem, how is slimr’s autocomplete implemented deserves a more detailed description. Given how R and Eidos are syntactically (and semantically) different, there are certainly limits at which autocompletion will not work. A description of these limitations should be included. For instance, many "live-during-simulation" objects (populations, individuals, genomes, etc.) are inspectable in SLiMgui, down to the autocompletion of their OOP methods and their arguments. Am I right to assume this sort of thing will not work in slimr, given that all slimr code is writte prior to a simulation run and so code analysis has limitations? Please write a detailed description how does the autocompletion on SLiM objects in R work.*

We have now fully explicated the autocomplete system, which we realize, is somewhat non-intuitive at first glance. We added a section to the manuscript just to explain it (lines ??-??). We also added a new, less verbose way of accomplishing the same thing, which we also explain.

*Fig. 1c shows how function slim\_output\_genlight can be used to extract some genetic data. When I try to type this into my loaded R session with slimr, I get an "object not found" R error. Did the authors rename the function or is it no longer provided by slimr? [Also, please note that "(c)" does not have a description in the Fig 1. caption text, although "(a)" and "(b)" do].*

Again, ongoing development led to us changing slim\_output\_ prefixed functions to slim\_extract\_ prefixed functions, in order to better disambiguate them from the slimr verb slimr\_output. Although slimr\_output has now been renamed r\_output (though slimr\_output remains as an alias for backwards compatibility). Figure 1 now only has 2 panels.

*Line 180: In the version of slimr which I downloaded on 2022-05-24, I see three slimr addins in RStudio: two which can copy "original code" to clipboard or into a new script file, and then a third addin (arguably the most useful one) which can open a slimr script in SLiMgui in its "native" rendered SLiM form. I tested it on the recipe 4.1 from the SLiM manual but did not succeed. Regardless of what I did, I kept getting "Error Executing Addin - R code execution error" from RStudio by running the addin, accompanied by "Error in inherits(x, "character") : argument "x" is missing, with no default" in the R console. I admit that I could have missed something obvious, but perhaps an extra sentence or two in the manuscript describing what these addins do and a description on how to use them how would be helpful. Either way, even if I did misunderstand how the addin works, reporting two dramatic errors is not the best way of handling UI issues. A message explaining how to use the adding is the right solution to this problem.*

Again, apologies for this. This was due to these features being too immature. For these revisions we now consider the plugins as experimental features and have removed them from the manuscript. They simply needed too much work to make them robust. We do hope to get them in good working order within the near future as we do think they could be very useful.

*## Example: "Data Input/Output"*

*The link in Fig. 2 caption is broken (leads to* [*https://rdinnager.github.io/slimr/articles/simple\_example.html*](https://rdinnager.github.io/slimr/articles/simple_example.html) *which gives 404). I think managed to find the correct URL eventually, so the rest of my comments are based on the HTML link I found here* [*https://rdinnager.github.io/slimr/articles/articles/simple\_example\_using\_migration\_and\_fst.html*](https://rdinnager.github.io/slimr/articles/articles/simple_example_using_migration_and_fst.html) *(assuming this is what the authors intended to link to).*

Apologies for the incorrect link. You found the correct vignette however. We have fixed the URL in the caption to point to the correct vignette.

*The vignette containing the example (URL above) relies on R package dartR, which has been removed from CRAN (see here:* [*https://cran.r-project.org/web/packages/dartR/index.html*](https://cran.r-project.org/web/packages/dartR/index.html)*). As such, only the part of the vignette that does not rely on results produced by this package can be run, I am not able to verify the rest. Please update the vignette so that it uses a package which can still be installed via standard CRAN distribution.*

dartR is now back on CRAN, so hopefully the vignette can now be run without issue.

*A clear indication of where in the rendered vignette can the exact code example from Fig. 2a be found is required. Perhaps using a dedicated heading in the vignette (i.e. "Code chunk from Example xyz in the paper")? For instance, there are several slimr code chunks, one of which contains a line "lines = readFile("~/Desktop/migration.*

*csv");" which certainly won’t work on the reader’s machine. It should not be necessary to scroll through the vignette to look for the exact code shown in Fig. 2.*

We have provided a link in the caption to the vignette as rendered on the slimr website.

*I think that a concrete example of how a statistic can be calculated during a simulation run using the slimr\_output "verb" and returned/accessed as a data.frame should be included in the Fig. 2a code example. This is a use case which would be of very high interest to R/SLiM users and it’s a bit of a shame that it’s not demonstrated in code. For instance, showing how to calculate a population genetic statistic of interest by a user-defined callback function utilizing the `do\_every` argument as mentioned by the authors would be useful (line 228). The sim.outputFull() slimr\_output call is clear, but the live tracking of a statistic and exporting it as a data.frame is the really interesting feature here. Just saying (line 227) "Output can even be accessed live during the simulation run via the use of callback functions. A do\_every argument tells slimr\_output not to output every time it is called, but rather only after every do\_every generations." does not really tell me what to do.*

The long vignette included as supporting information for this manuscript contains a detailed example of using slimr\_output (now r\_output). Accessing data live while the simulation is running is demonstrated in a vignette included in the package, which we have now included a link to in the text.

*## Example: Simulating Nucleotide Evolution*

*Line 295 and onward: Again, I strongly suggest that the authors put the complete, reproducible example code in a separate vignette (code and figures included). When I tried to copy the code example from the PDF to my R session (one tiny bit of code at a time to make sure the PDF linebreaks didn’t break the R syntax), I kept getting syntax errors from R. Only when I carefully copied the entire script specifically line by line (something which I don’t think many users would be willing to do) I managed to run the code the way authors intended.*

Done

*When executing `slim\_results\_to\_data(results)` (line 356) I received the following error message (below). I’m not sure whether the error comes from slimr or from R itself (I don’t use Bioconductor in my own work so I don’t recognize the message) but I suspect that Biostrings should be added to the list of explicit dependencies in the DESCRIPTION file to prevent this error from happening. That way (as I suggest in a comment above), the whole example including figures will be render-able automatically on slimr’s pkgdown website.*

*Error in `assert\_package()`:*

*! This function requires the Biostrings package, which is not installed. Install with BiocManager::install("Biostrings").*

*Run `rlang::last\_error()` to see where the error occurred.*

Biostrings is in the Suggests section of DESCRIPTION and so does get installed and built automatically by pkgdown. All packages that are not required for the main functionality of slimr (that is, writing SLiM code and running them) are put in Suggests in order to keep the number of hard dependencies of slimr manageable. The error message you received is what you see when you use a slimr function that relies on one of these non-essential packages, so this is the expected behaviour. The example is now a rendered vignette on the package website.

## Example: Scientific Hypothesis Exploration Example

*I’m afraid I have to say it again — I strongly suggest to provide a clean vignette available in slimr’s documentation which will be runnable with a minimum amount of additional text and figures and additional code. The reader shouldn’t have to be forced to scroll through more than 50 pages long(!) rendered HTML file to find a code block to run the code from Fig. 5a. I think that adding a long supplementary notebook with much more extra information about the scientific background, maps, and analyses is perfectly fine, but most of it is not (and should not be) needed to understand the functionality of slimr presented in the code in Fig. 5a. Boiling the vignette to the bare essentials will make it easier for the reader to test the code and see the power of slimr in action.*

*This leads me to a bigger problem: even after opening the provided HTML file supplementary notebook, I didn’t succeed in running the code at all. The first code chunk contains the line:*

*gen <- read\_rds("data/herm.rdata")*

*The second code chunk contains:*

*abund <- read\_csv("data/mammal\_captures.csv")*

*But where can I find those files? They are not part of the supplementary data with the submitted manuscript and — as far as I can tell — they are not distributed with the R package itself (why not if they easily can be?).*

*I strongly believe that a paper which describes a new software library should provide fully reproducible code examples which can be run with minimal effort on the user’s part, simply by using available documentation (i.e. R vignettes), including all code and data used in the manuscript. I recommend distributing all necessary data together with the R package (*[*https://r-pkgs.org/data.html*](https://r-pkgs.org/data.html)*) which will solve the issue completely and will allow — yet again — the distribution of the fully reproducible vignette together with the R package.*

We agree and have made the data available from within the slimr package itself. The main example vignette is now self-contained and it should be possible for anyone to run it as long as slimr is installed.

*As such, I’m sad to say that I haven’t been able to run the code behind this example at all, and therefore can’t properly review it. This is a shame because authors themselves say (line 426) that it represents "the main slimr example from this manuscript".*

This was indeed too bad, and we apologize for the oversights. We hope we have remedied the situation for any future reviews you may accept to do for this package!

*## ABC Results*

*Again, I don’t think there is a technical reason why this example should be distributed by a plain R script rather than a small, fully reproducible R vignette available on the website as standard documentation, including the code but also visualizations (here shown in Fig. 7 and Fig. 8). Regardless, because the provided example script appears to rely on outputs from the large HTML notebook file dedicated to the previous example (as indicated by the very first commands such as `read\_rds("data/pop\_abunds.rds")`), I was not able to run the code and as such can’t review this section of the paper at all because it's not reproducible (missing input data).*

We've included the data files necessary to run the script in a supplementary zip file that we will post to figshare should the paper be accepted.

*Page 23-25 appear to discuss general aspects of ABC inference and the text does not actually involve slimr-specific issues at all. These would be all applicable even if a researcher would use any other standard method for doing ABC inference. I suggest shortening this section (or adding this material to a vignette) and maybe investing the valuable "page real estate" to slimr-specific issues, some of which I highlighted elsewhere in this review (autocompletion discussion, Eidos-R type coercion, etc). I do appreciate the cool research facilitated by slimr here (despite not being able to run the code in question) but these three pages read more like a section from a research paper, not a paper describing the software library slimr. I think this extended discussion of ABC results should be moved to a vignette dedicated to this example.*

We've shortened this section, moving much of it to supplementary material.

*## slimrmodels (lines 514-540)*

*This section left me a very confused. The authors write (line 516) that they "have developed a companion R package called slimrmodels (<https://github.com/rdinnager/slimrmodels>), in which we have implemented several potentially useful simulation models". However, the GitHub repository contains only a stub git repository with basically no code in it. There are certainly no "several potentially useful simulation models" provided by the authors here (state of the repository at the time of writing this review:* [*https://github.com/rdinnager/slimrmodels/tree/dba32c9b50c1eed7ac063bb9371677f24623782b*](https://github.com/rdinnager/slimrmodels/tree/dba32c9b50c1eed7ac063bb9371677f24623782b)*).*

*Similarly, the authors write (line 538) that "More information about slimrmodels can be found in the documentation for the package". But the GitHub repository contains no documentation whatsoever? There’s no man/ directory and the landing page contains the text "Example — This is a basic example" with no further information.*

*Why is an entire paragraph of the manuscript dedicated to an R package that — despite the section implying otherwise — has almost no content? Indeed, the code presented on lines 529-537 can’t be run at all, even though I did install slimrmodels on my machine.*

*I am not sure what the intention was behind including this section but unless the promised functionality of slimrmodels is actually provided, the whole section needs to go. Otherwise this all will look extremely suspicious to the potential reader.*

We agree this was a major problem and apologize for this. The slimrmodels package was an ambitious idea but whose finishing touches got forgotten in the final rush of finishing and submitting the manuscript. The slimrmodels package was working locally but it was not pushed to github, and so was never available to others. There is no good excuse for this, we can only apologize. The slimrmodels repository still requires significant polish to make it ready for general use and so we have decided to remove this section. Once completed and polished, slimrmodels will be released at a later date with its own accompanying software note.

*## Other major comments not related to code examples*

*In several places (lines 75 and 114) the author say that "slimr […] mimics the syntax of SLiM" and that slimr code is "very similar to the native SliM [sic] syntax, with a few modifications". On line 172 they write "slimr makes it easy to write SliM code in R after the user learns a few differences between SliM and slimr". But what are those differences, exactly? The paper currently does not contain a detailed description under which do slimr and SLiM/Eidos differ (syntactically and semantically).*

*Similarly, on line 218 the authors write: "slimr\_inline automatically detects the type of R object and attempts to coerce it into a format compatible with SLiM. Currently supported types are all atomic vectors, matrices, arrays, and Raster\* objects […]" How does slimr handle cases in which Eidos and R are not fully compatible? For instance, Eidos appears to be much stricter in handling integer vs float values than R. Similarly, missing values are also handled differently. A table indicating how types are explicitly coerced from R->SLiM and SLiM->R by slimr would be helpful. The fact that Eidos looks almost like R but sometimes behaves in surprisingly non-R ways routinely trips up even advanced R & SLiM users.*

We agree this is underdeveloped currently, and improving this will be a focus of further development of slimr. The language we used in the manuscript was not very precise. We should have said that it detected the class of R objects (not type), that is, vector vs. matrix or array, or a special object class like Raster\*. Currently, slimr does not have a sophisticated type detection for converting to Eidos types. slimr\_inline currently uses a very simple and naive method for getting objects into SLiM, which is to literally insert them as text into the script. This means that they will get whatever type that Eidos detects them to be from their input format as plain text. We think in most cases this should work fine, especially since R users don't tend to worry too much about explicit types in day to day coding. However, we do think it would be good to have something more explicit for more advanced use. But we have not implemented this yet. It is on the roadmap for slimr version 1.0.

*A short paragraph describing what exactly happens behind the scenes of slimr’s advanced concepts (inlining of objects into SLiM, templating, etc) would be helpful. At which point does code generation or templating happen? Is it done by simple string substitution on code-generated SLiM scripts? These features of slimr are extremely interesting but they are currently presented as black boxes which is not ideal.*

We have added a paragraph about how slimr\_inline (now r\_inline) works under the hood and a table describing the major differences in syntax between slimr and SLiM (e.g. how to convert back and forth).

*# Minor comments*

*Too often the authors write SLiM as "SliM", RStudio as "Rstudio". The correct spelling is SLiM and RStudio.*

This has been fixed.

*Line 186: The authors mention a slimr\_script\_coll object, which can potentially contain a list of slimr\_script objects. It would be helpful to add a sentence telling the reader what "list of slimr\_script objects" means in this context, and that this will be explained later in the paper.*

Done

*Line 280: "[…] the use of slimr <ON> a short and simple example […]" — missing the word "on".*

Done

*Line 290: "The forcing operator tells R to evaluate what comes after first, […]" After first what? A word appears to be missing?*

Just an awkward phrasing that has now been fixed. What was meant is that the operator tells R to evaluate the following expression first, that is, before anything else in the syntax tree it is part of. R by default will only evaluate an expression when it is actually used, which could be deep in the syntax tree. In other words, the expression gets evaluated and then the result of the evaluation gets passed into the function it is being used in, instead of it being evaluated when it is used where it would be treated as SLiM code instead of what it represents within R. We've added a few more sentences of explanation to clarify in the main text.

*Line 393: Did the authors have in mind Figs. 5 and 6?*

Yes, this has been fixed.

*Line 511: "[…] slimr […] allows open development of simulations by developers with much less experience with R coding, and requiring far less time." Did authors by any chance intend to say "much less experience with \*SLiM\* coding" (i.e. not R coding)? I would argue — given the advanced metaprogramming concepts, inlining R code in SLiM, templating — that the amount of knowledge of R (or programming in general) is actually rather significant. In my experience, most R users have no idea how metaprogramming in R works and what significant portion of R library code uses it (the entirety of dplyr/ggplot/etc, for instances). This is of course subjective, but I’m nearly certain this sentence should say SLiM and not R.*

We have reworded. You are correct on this point.

*In number of places, the authors write that slimr represents an interface to "SLiM 3.0" — is the version number to be taken literally? SLiM has a relatively frequent release schedule where new features and functions are added sometimes (but not often) in a backwards incompatible way. Does slimr follow the SLiM release schedule? In any case, specifying the exact minimum SLiM version supported by slimr is important to be able to judge whether a SLiM feature the user intends to use is available in slimr (recent additions such as DataFrames etc are particularly relevant here — are these supported during in the SLiM <---> R conversion?).*

To be honest I didn't realize that DataFrames were supported now in SLiM, and this seems like an obvious area for integration between R and SLiM. Supporting DataFrames has been added to the milestones for slimr version 1.0. We have changed references to 3.0 to >3.0. The hope is to maintain compatibility between slimr and any version greater than 3.0. This so far seems to be possible since our upgrades to support SLiM 4.0 did not break SLiM 3.0 and 3.5 scripts (thanks in large part to the efforts of the SLiM developers to maintain backwards compatibility themselves). This may become impossible at some point but for now >3.0 is accurate. The only exception to this is in the documentation, for which we only include the latest SLiM version documentation, as it proved too complicated to try and maintain multiple versions of documentation within the package.

*Unit tests - As with any R-X interface package which involves translation of code between R and X, it is important to show that the conversion of R code works exactly as intended. In this context, unit tests which would run a pure SLiM version of some SLiM recipes, then their manual slimr translation, making sure that both produce the exact same result would significantly increase confidence in the software. Does slimr include such unit tests? How much SLiM functionality do they cover? I think dedicating a couple of sentences to such tests of accuracy/reliability would significantly help the readers in gaining more confidence in slimr.*

This is a great idea but we currently do not do this. We have put it as a milestone for version 1.0 since it will require some substantial testing infrastructure development to get it up and running, but this will be a high priority. Thank you for the suggestion!

*This is possibly a very minor nitpick: how is presence/absence of terminal semicolons handled by slimr? Obviously R doesn’t require semicolons but Eidos does. Yet semicolons are sometimes missing after a slimr statement in code examples (lines 321, 329, etc.) and sometimes are present (lines 308-314, etc). I assume semicolons are completely redundant in slimr, but even ignoring the question of aesthetics, a unification of style would be less confusing to new users.*

Good point. You surmise correctly that semicolons in slimr are effectively ignored because they are automatically added at the end of each line during translation to SLiM code even if they are not present in the R code. We thought it might be good practice to include them in slimr code so as to not develop bad habits when going back and forth between slimr and SLiM development as some users may wish to do. However, because they do not matter in slimr code it is easy to forget to add them, since this results in no consequences. We have unified the style of code examples in the manuscript, and we have noted this difference in the manuscript in our new table of SLiM - R differences (Table 1).

*In some cases, the author’s write that "the user can do" various things, yet in other paragraphs they write that "you can do", speaking to the reader directly. This is quite awkward, in particular the paragraph starting from line 112 reads a little strangely.*

We've tried to be consistent with this in the revised manuscript, using 'you' to speak directly to the reader..