We thank the anonymous reviewer for their comments, and for taking the time to review the submitted manuscript. Responses to each submitted comment are provided below, and we plan to make changes to the manuscript in response, as indicated in the author comments.

Original review comments are provided in *italics*, and author responses to the comments are in **bold** font in the sections below.

0.1 General comments

This manuscript provides a description of a set of R functions to process in- and output files and the contained data, for the purpose of running the hydrological modelling software Raven, which itself is available as a C++ executable.

The manuscript is generally well-presented and well written, and I have very few specific comments. However, I make the following more general observations:

1. *I am actually not sure whether the manuscript fits any of the designated manuscript type. I suppose that it is classified as a development paper because of the ample references to reproducibility in the manuscript. However, the original model is available as an open source code as well and therefore perfectly reproducible (at least in the sense that it is described on the about GMD web page). So at most, it is an enhancement of the usability of a specific model rather than its reproducibility.*

   The manuscript type was changed from its original submission to a development paper by the handling editor, so we respectfully suggest that this is the best categorization of the manuscript type, based on the discussion of technical aspects of running models and reproducibility of results.

2. *The other reviewer has made some very useful comments on the presentation, with which I fully agree. Overall, I think that the manuscript is too wordy and can be reduced substantially. Specifically, the authors seem to be at pains to convince the reader about the importance of open source software, accessibility, and good practices in model development. I don't think that the GMD readership needs such advocacy. It distracts from the core message and makes the manuscript unnecessarily long and somewhat tedious to read. (For example, the section L.137 - 146 is quite trivial and may be deleted entirely, but also many other sections can be streamlined).*

   Some sections of the manuscript will be reduced in the revised version based on the comments of both reviewers, including adjustments to L. 137-146 as suggested.

3. *The technical implementation of the package is quite straightforward, and does not make optimal use of advanced functionality of R.*

   Author responses are provided to the specific comments in the following section on Technical Comments.

0.2 Technical Comments

Specifically:

1. *The fact that the model needs to be run separately is not very elegant. It would be ideal if the Raven model itself is distributed with the package as a dynamic library, and can be loaded as such by the R process. This would avoid the need for separate installation of the model, as well as the slightly clunky way that the executable is called by the rvn_run() function. It would also help with the next point.*

   This point is brought up by reviewer 1 as well, and although it may be theoretically possible to include the entire Raven project in the RavenR distribution and compile it through the Rcpp library in R (or similar approach), there are at least two technical reasons why this is not ideal aside from the fact that this is not the stated purpose of the package. First, the size and complexity of the existing Raven code would necessitate a massive undertaking to import the project and ensure it can compile in R. Second, the Raven framework is used

by a variety of users, and this could create technical issues for users that have their own (non-RavenR) workflows and forecasting environments with Raven, such as those running Raven with direct shell interaction on clusters or with python tools (e.g., using the RavenPy API). Keeping a copy of the code in RavenR as a non-master version may be possible, but would lead to code management duplication and additional overhead. Installation of Raven is trivial, as it simply requires the presence of the self-contained executable file. The functions to run the executable operate similar to other well-known hydrologic modeling packages like flopy [Bakker et al., 2016]. However, improved integration with RavenR and other scripting languages is a worthwhile future goal for Raven that is under development.

2. *The fact that the scripts writes the input files to disk, which are then subsequently read by the executable (and vice versa for the output files) is inelegant at least, and probably also inefficient as well. If the model itself were implemented as a dynamic library then the in- and output data could be passed in memory to the model, which would greatly enhance performance in use cases such as monte carlo simulation.*

   The authors agree that this would certainly be more efficient for projects requiring many model runs, and that this is a computational limitation of the current software implementation. However, the effort to convert Raven into a dynamic library is significant and well outside of the scope of this paper. For such work, other tools and scripts with less overhead may be preferable. Many of the use cases discussed in the paper relate to preparing input files and analysing final output files, which would not suffer the same limitations as described above. We have also used RavenR successfully to process the outputs from tens of thousands of calibration experiments, and found the computational demand of this read-write process to be acceptable. Raven does include comprehensive features for controlling the frequency and extent of output files, which can greatly help in keeping voluminous sensitivity/uncertainty exploration run times to a minimum.

3. *The package makes relative limited use of the object oriented nature of R. It does use relevant classes such as xts and lubridate, but does not define any classes itself. This results in a very long list of functions, essentially one function for every step in the analysis. It would be much more elegant (and efficient) to define a set of classes (e.g., one for each in- and output file, by extending classes such as xts) and then use method dispatch to read and write them, as well as any other standard processing such as aggregation. This would reduce the need for a long list of different functions to a few read() and write() commands, and allow for method dispatch on existing xts functions.*

   We agree that this would be a more efficient and elegant way to organize the package and make use of the more advanced aspects in R as mentioned, but by no means necessary to provide access to useful functionality. This will be strongly considered in a major version update for future package revisions.

Lastly, while the examples in the manuscript are generally easily reproducible, some of the examples in the online documentation are not, for example because they include idiosyncratic path statements. i strongly recommend the authors to read through R guidelines such as the ones below, and cross-check that all the code adheres to these good practices:

https://www.tidyverse.org/blog/2017/12/workflow-vs-script/

https://www.carlboettiger.info/2013/06/13/what-I-look-for-in-software-papers.html

The RavenR package is compliant with the rather rigorous standards of CRAN, and the examples are being continually updated for clarity. The authors will perform a search for idiosyncratic path statements and revise them based on the standards linked above in future versions of the software. We thank the reviewer for directing us to these resources.

0.3 Conclusion

To conclude, I believe that this is certainly a useful piece of software, however for me the manuscript reads too much like a manual instead of a scientific paper, even of the type that GMD aims at. I think that there is scope for streamlining, and ideally going a bit beyond simply presenting a wrapper, towards exploring how even something as simple as a wrapper can incorporate state-of-the-art software design concepts. This does not

mean that the software needs to be entirely implemented according to the recommendations above. But some attempt, or at least a discussion as to why this may be scentifically non-trivial, would lift the scientific value of the manuscript in my opinion.

We will certainly streamline the revised version of the manuscript based on the recommendations of the reviewers, and will include some notes in section 2.1 and 2.2.1 to make clear the interplay between Raven and RavenR, and discuss why including the Raven source code in the RavenR package directly would not be an ideal solution. We hope this will alleviate some of the concerns with that aspect of the software. We note here that the focus of this manuscript (and our development of RavenR) is not on incorporating state-of-the-art software design, but rather providing a software contribution which benefits the state-of-the-practice.

References

M. Bakker, V. Post, C. D. Langevin, J. D. Hughes, J. T. White, J. J. Starn, and M. N. Fienen. Scripting modflow model development using python and flopy. Groundwater, 54(5):733–739, 2016. doi: https://doi.org/10.1111/gwat.12413. URL `https://ngwa.onlinelibrary.wiley.com/doi/abs/10.1111/gwat.12413`.