Dear editor,

We are very grateful for the invaluable comments and suggestions by the reviewers, which have significantly contributed to enhance our software and our manuscript. We have now produced an improved version of the DnoisE software, which not only fixed all bugs and issues pointed out by the reviewers, but is also equipped with more customizable options, making DnoisE a more versatile tool. We are submitting a new version of our article, including a detailed analysis of a simulated dataset of error-affected sequences, which we think will help to clarify the comparison between the results of our new software and Unoise. Our answers to all reviewer's comments are detailed below. We hope that this version meets the standards to be published in PeerJ. Thank you very much for your consideration,

Owen S. Wangensteen (in behalf of all authors)

Comments from the reviewers

Reviewer: Andrea Telatin

Basic reporting

I would like to thank the authors for the improvements and for kindly addressing my questions.

I recommend a more detailed explanation of the core algorithm (as schematized in Figure 1), and a clearer comparison (in a dedicated paragraph) with the procedure previously published in Antich 2021 and UNOISE itself.

I think added clarity in this aspect is pivotal for a better understanding of the paper, especially for those who already implemented the methodology as originally described.

In particular, I recommend not to refer to the software used in Antich 2021 as "beta", as it was in fact used in production for the paper itself, and according to the authors, the changes are not expected to produce different findings.

The version used in Antich et al (2021) was indeed based on the same algorithms and the results are not expected to differ. However, that was a much less mature version, with few parameterization options and not user friendly. This is now explained in the text (first paragraph of "DnoisE performance section"). In addition, there were several bugs that have been corrected and many new features (f.i., minimal read abundance filtering, sequence length filtering) and new sources (f.i., the Levenshtein distance module) have been added. Antich et al. (2021) only used the software and did not compare it with Unoise in a quantitative way, which is now presented in the manuscript as described below. There was also no explanation of the program workflow and processing (Figs 1 and 2 of the present manuscript). So we feel that, albeit the core functions were basically the same, a paper was required to present the software properly and make it available to a wider audience.

Validity of the findings

In the "DnoisE performance" paragraph I suggest a more explicit comparison with Unoise. A Venn diagram with the shared identical sequences will be a good addition.

The mention of future work with mock communities is sound and appropriate, yet the increased number in ESV detected with entropy correction can include false positives. Considering that the method was applied previously, and the entropy correction is the unique feature of the presented tool, I think that a formal validation of the entropy-correction algorithm is required, at least with simulated reads comparing the effect of the correction and of naive approaches.

We thank the reviewer for this suggestion. It is true that a mock community of true haplotypes would be a good option to validate DnoisE but building such community with the required complexity would be a cumbersome endeavour. Following this comment we have now added a comparative analysis with a simulated dataset of "good sequences" affected by random errors, following Turon et al. (2020), including tracking from which “true” sequence each simulated error amplicon derived. We have changed the structure of the “DnoisE performance” section to show the results of this comparison, adding a “merging performance” heading, and incorporated two new figures to the ms.

Note also that we are now referring to the algorithm presented by Edgar (2016) as the UNOISE algorithm, to distinguish it from the software that applies it (UNOISE3 being the current version).

As UNOISE3 is not open source, has a chimera filtering procedure embedded, and is based on a posterior assignment of reads using -otutab, we could not directly compare the two programs. Instead, we first compared our DnoisE without entropy correction (which uses the same formula of Edgar 2016) with UNOISE3 by picking the information on sequence merging from the -ampout and -tabbedout files generated by UNOISE3, and without an -otutab step. We could verify that both programs produced the same results (99.99% of ESVs coincident involving 99.999% or reads). Once validated, we went on using DnoisE with and without entropy correction, to evaluate the entropy-correction algorithm at different alpha values, as our software is more flexible and allowed us to program features such as the tracking of original sequences.

Using this simulation we could compare not only between the results retrieved using or not entropy correction in terms of true and false positives but also between the different merging criteria that DnoisE can perform. This simulation showed clearly the potential of DnoisE over UNOISE3 and, particularly, the usefulness of its flexibility in choosing the adequate denoising strategy for a given dataset. This exercise also led us to add the --min_abund parameter to DnoisE which we think is a useful option not seen in other software.

The simulation dataset consists of 1,000 sequences considered error-free and with realistic abundance distribution, referred to as ‘original’ sequences, obtained from marine communities in Turon et al (2020). With this dataset we simulated sequencing errors, at

0.005 error rate (datum taken from the literature as representative), and obtained 265,297 error sequences. For the generated sequences (‘error sequences’ hereinafter) we kept in their ids the information of which was the original sequence from which they derived. With this information we could calculate the amount of error sequences that are denoised and if they were merged with their “true” original sequence or not.

As shown in results, entropy correction allows to retain more ‘good’ sequences but also some ‘error’ amplicons. We also suggest a filter of minimum abundance of 10 reads which eliminates most of the retained error sequences. To perform this final filter we have enabled an abundance filtering option (-r --min_abund; set as zero as default) to specify the minimum amount of reads required to retain a sequence.

Reviewer: Niklas Noll

At first, I was very happy to see the improvements of DnoisE and I think the authors did in general a great job addressing our concerns.
However, I have noticed a few things that I would like to describe:
I tried DnoisE with the provided example files and also with my own data. Using my own data, there was no difference between the output if entropy correction was enabled (-y) and if not. It resulted in the same sequences. This seems to me as a strange result and I wonder how this could happen? I would offer to provide my test data. If needed, please contact me via email: n.noll@leibniz-zfmk.de
The example from your github repository did however show different results. But not as expected. Here we find less sequences if entropy correction was enabled. This is contrary to the findings of Antich et. al 2021 and I wonder if this is due to the different dataset or is it because of a change in the entropy calculation?

We sincerely thank the reviewer for his thorough job testing our software. During the restructuring of the software for the last revision, entropy correction was inadvertently disabled (!). This has been fixed now. We have also detected other bugs that have been solved.

An important change that we performed in the last version was a filtering of sequence length when using entropy correction. This filtering step is necessary because entropy correction can only be performed to aligned sequences of the same length. Without entropy correction this is not necessary as the Levenshtein distance can be calculated even if there are indels. We also wanted to accommodate the natural variability in sequence length of some markers. This variability, for coding genes, is usually due to indels of one or several codons, and thus length differences are multiple of three nucleotides. With our COI fragment, in our experience with several datasets, ca. 95% of all sequences obtained have the modal length of 313 (modal length is customizable), but there is some variability among different species. DnoisE with entropy correction thus accepts sequences of (modal length) ± 3*n, being n the number of codons of difference. All other sequence lengths are considered erroneous and the corresponding reads are eliminated. This procedure is now explained in the ms (in the paragraph before “Parallel processing”).

The length filter explains the effect detected by the reviewer with the DnoisE-test dataset. This dataset included sequences of different lengths. There were 14,073 sequences of which 13,291 had 313 bp, and 148 more had “compatible” lengths (313 ± 3*n). On the other hand, 1,233 sequences had unaccepted lengths and were removed when entropy correction was performed. This results in 1,703 ESVs without entropy correction and 1,785 with it. The increase in ESVs obtained as a result of entropy correction is almost compensated by the fact that, without entropy correction, 1,233 more sequences of unacceptable lengths are included. For large datasets the effect is negligible and more ESVs are retained with entropy correction. Note that in Antich et al 2021 we used a test dataset of sequences of uniform length (313 bp) and therefore the length filter was not necessary. It has only been implemented in the present version. We now explain in the README.md file of GitHub the potential effect of excluding sequences, and we have also added another test dataset including only the 13,291 sequences with 313 bp. So, the interested user can compare the results.

After I installed the program with ./install.sh the binary did not work with the following error message:
./DnoisE -h
Traceback (most recent call last):
File "src/DnoisE.py", line 12, in <module>
ModuleNotFoundError: No module named 'denoise_functions'
[15164] Failed to execute script 'DnoisE' due to unhandled exception!
It does however work if I use python3 ./src/DnoisE.py. Therefore, I suggest fixing the issue with the binary.

In the reviewed version, the binary was supposed to be produced by pyinstaller, however we have now changed it to nuitka (https://nuitka.net/). It takes more time but produces less errors. We have now also changed the module used for the computation of the Levenshtein distance,which was the origin of many problems in different systems. We now use the module from maxbachmann (https://github.com/maxbachmann/Levenshtein) instead of the one from ztane (https://github.com/ztane/python-Levenshtein). The binary produced is ./scr/DnoisE.bin and the installer should now work in most systems.

The result output *denoising_info.csv seems as if it is not correctly generated. Another problem is that the output files are neither explained in the current manuscript and also not on the github page. I would expect a brief explanation of what every column name means. For example xavier_criteria.
The second column “mother_d” seems to be the name of the mother sequence, whereas d shows otherwise an integer.
The column “mother_ratio” holds the name of the mother sequence and is not a ratio

The reviewer is right, we have now updated the info.csv column names and we describe this output file in detail in the Github README.md

The info.csv returns information on how the sequences have been merged. ‘Mother’ sequences have only its id name with no other information. For each ‘daughter’ sequence the following information is given.

‘daughter’ -> the sequence from which the information is retrieved

‘mother_d -> the mother sequence corresponding to the ‘d’ criteria

‘d’ -> the d value corresponding to the comparison between daughter and mother_d

‘mother_ratio -> the mother sequence corresponding to the ‘ratio’ criterion

‘ratio’ -> the ratio value corresponding to the comparison between daughter and mother_ratio

‘mother_ratio_d -> the mother sequence corresponding to the ‘ratio_d’ criterion

‘ratio_d’ -> the ratio_d value corresponding to the comparison between daughter and mother_ratio_d

If a entropy correction is performed, the following information is also given:

‘difpos1’ -> number of differences in position 1 of the codon corresponding to the mother_ratio_d comparison

‘difpos2’ -> number of differences in position 2 of the codon corresponding to the mother_ratio_d comparison

‘difpos3’ -> number of differences in position 3 of the codon corresponding to the mother_ratio_d comparison

‘dtotal’ -> number of total differences corresponding to the mother_ratio_d comparison

‘betacorr’ -> beta value corrected by entropy values corresponding to the mother_d comparison.

The latter value is used during the process to choose the mother_d sequence and we think that it could be useful if future changes on the correction formula are performed by users.

The help message displays:
y --entropy_correction a distance correction based on entropy is performed (see ENTROPY CORRECTION below). If set to F, no correction for entropy is performed (corresponding to the standard Unoise formulation)
There is no “below” and the entropy correction is not explained further. It is also misleading that no correction is performed if -y is set to F. The default of DnoisE is to not use entropy correction. Using the -y option enables it. Since this seems to be just a flag argument, it cannot be set to false and it would also be unnecessary.

There was a mistake in the --help message. It is true that it is just a flag argument. It is now solved.

If the input file is of type fasta, a semicolon is needed at the end of the header. Since the

output files from usearch or vsearch do not include semicolons at the end, I would recommend allowing users to use files without these.

Solved

I could still find no automated tests that ensure the correctness of core functions or output results.
We have now created the ./src/test_DnoisE.py script that calls .json files to run a small dataset for the core functions and to compare with expected results. It can be run from the terminal as follows:

python3 -m unittest test_DnoisE.py

L106-L111: It should be added that aligned sequences are needed. The entropy correction is the main novel feature of DnoisE and using it without aligned sequences this feature does not work properly.

As detailed above, DnoisE is now able to work with sequences of different lengths, and it will automatically remove sequences of unacceptable lengths. Details about the sequence length requirements are now better explained in the ms.

Note that, following a request from the first reviewer, the ms includes now a simulation study to compare the performance of the program with and without entropy correction (the latter equivalent to the UNOISE3 software) in terms of good sequences recovered, false positives, and correctness of the merging criteria. We have changed the structure of the “DnoisE performance” section accordingly, adding a “Merging performance” heading, and incorporated two new figures to the ms.

Note also that we are now referring to the algorithm presented by Edgar (2016) as the UNOISE algorithm, to distinguish it from the software that applies it (UNOISE3 being the current version).