Geoscientific Model Development Discussions

Geosci. Model Dev. Discuss., referee comment RC3
https://doi.org/10.5194/gmd-2020-391-RC3, 2021

Comment on gmd-2020-391

Matt Hall (Referee)

Referee comment on "*dh2loop* 1.0: an open-source Python library for automated processing and classification of geological logs" by Ranee Joshi et al., Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2020-391-RC3, 2021

Thank you for the opportunity to review this paper. It's good to see a case study applying open source tools to a common problem in subsurface — the extraction and regularization of data from open data stores. And it's fantastic to see reproducible research being produced and documented for the community. I wish more research in our field followed your excellent example.

I am recommending 'major' revision, but most of the revision that I believe the manuscript could benefit from is simply reducing the amount of content. That is, I don't think there's new work to do, or that the content needs many changes per se. (Though there are a couple of issues in the code that need to be sorted out.)

I have read the other reviews and agree with their comments. I don't think what I have to say is substantially different. In particular, I think we all agree that there is a useful and interesting contribution here.

General remarks

The paper seems long. The length, and difficulty of feeling out the structure of a long paper, probably led to a feeling of confusion about whether I was reading a geological case study, a paper on a new approach to ingesting data, or the documentation for a software package. Perhaps the manuscript is trying to do too much? I wonder if you could split it into pieces? For example, move the more prosaic stuff (lists of tables, etc) to the docs; describe the more technical language modeling piece in a short nerdy paper about that; and present the case study as a short geological success story?

If the authors and editors feel a long paper is justified, then my suggestion is to be extra careful about the outline of the paper, so that a person knows what aspect each chunk of the paper is addressing.

Compounding the length is that there's quite a bit of redundancy between text, figures and captions. Figure 9 is an archetypal example of this. I think the caption (which I cannot parse) essentially spells out the diagram (which I cannot read very well), and virtually all of the data is also written out in sections 3.1 to 3.3. My advice is to frame the ideas with the text and put all the details of table names and data in the figure, which should barely need a caption.

Some pieces seem better suited to the tool's documentation. Appendices B and C probably don't need to be part of a paper. Similarly, I think detailed descriptions of tables are clogging up the text and getting in the way of the reader. Standard measures like precision and recall, or string matching statistics, probably don't need to be described in detail.

Detailed remarks

- Title: Python should be written with an uppercase P.
- Abstract: I'm not sure you need the first paragraph; it's introductory material. If you want to motivate the problem, I think you can do it in a sentence (the second sentence captures it for me).
- Line 12: surveys, textual (no 'r')
- Line 15: hundreds of
- Line 19: replace 'that provides the functionality to extract...' with 'for extracting...'
- Line 23: 86%
- Line 51: You want spaces after all your semi-colons.
- Section 2: I suggest sticking to sentence case for your subheadings; at least make them all the same.
- Section 2.1, Conventions: The font is called Lucida, not Lucinda. I like the idea of a font convention, but if you're going to use one it is important to be consistent... and being consistent pretty hard. E.g. line 174: collar should be a table name, I think? Line 217: are these tables? Line 225: dh2loop should be italic.
- Lines 166–184: This paragraph is really hard to parse. I think the reader may start wondering if they need to know this information. If the info is important, I think you can let Figure 2 do the work.
- Line 280–295: Similarly, I think you're just describing Figure 4.
- Sections 2.4.1, 2.4.2, 2.4.3: This feels like documentation.
- Lines 419–440: There is a lot of information here; I feel it is perhaps best suited to documentation.
- Table 1: Not sure what to make of this. What do the checks and x's mean? What is the score? Why is one row bold?
- Sections 3.1, 3.2, 3.3: See my earlier remarks about these paragraphs vs Figure 9 and its caption.
- Table 2 and Figure 10: I find these hard to get insight from.
- Confusion matrices: There are a great many data tables here but I am unsure what to do with this information. I see that there are a lot of under-represented labels (low support). The only reference I could find to this problem mentions normalizing the

accuracy, but this doesn't really solve the problem, it just makes the confusion matrix colourbar fit better. So you're still unable to balance precision and recall (with small support, one of them is probably going to be bad). I'm not sure what you could do about it, other than get more data, but it might be worth mentioning.

Remarks about code

There are some problems with the licences on some of the code you are using.

QDriller (https://github.com/valheran/QDriller) is licenced under the GPL, which is a copyleft licence. This means that if you use it then your entire project must be licenced under the same (or a compatible) licence. **You are not allowed to use GPL'd code under an MIT licence.** So you have a few options:

- If you keep that code as-is, you must licence your entire project under the GPL.
- Ask the original author to grant you permission to re-use the classes you need under a more permissive licence. (Ideally, it would be, say, LGPL and a completely separate library that you could simply depend on, without copy/pasting the code across.)
- Rewrite this functionality from scratch.

GeoVectoLitho (https://github.com/IFuentesSR/GeoVectoLitho) is not licenced, as far as I can tell. Normally that would mean that you cannot use it without written permission granting you some rights. So make sure you have written permission to place it under your MIT licence, or you could ask the authors if they will consider an open licence. At the very least, you need to make it clear that the mlp.py module is based on work that is copyright of those authors and explain the terms under which you're using it. You should probably also exclude it from the MIT licence so that nobody comes along and picks ut up thinking it's FOSS.

In each of these cases, if you cannot find a way to

The Where to start instructions on the repo need VTK, folium and ipyleaflet adding to the installs. You could just tell people to install everything in requirements.txt with pip install -r requirements.txt. After that the notebooks ran for me (though I didn't run them all right through).

In Notebook 0, the map did not appear; I didn't try to figure it out. Enabling the extension didn't help. (I'm on a Mac, Chrome browser.)

I'm not sure these comments should form part of a peer review. If you don't plan to maintain this library into the future, you can probably just ignore all this. But in case it's useful:

- I strongly recommend writing docstrings for all of your functions.
- You should write tests for your functions. "Untested code is broken code."
- You can remove components of the standard library — sys, math, re, time, itertools, etc — from requirements.txt. These are included in every Python installation.
- I advise against hard-coding the file encoding in your functions (with encoding = "ISO-8859-1") because this seems like something another person is quite likely to want to change, e.g. if they have a UTF-8 encoded file. You should expose it as an argument. The same goes for database connections and other things another user might want to change.
- You should delete unused code, rather than commenting it.
- I'm sure you know about geopandas already, but it seems like it would be useful in your project.
- Note that PyProj emits a warning in Python 3.8+ FutureWarning: '+init=:' syntax is deprecated.
- I don't know if psycopg2 connections or cursors are compatible with Python's context manager (in which case you should 'with' blocks for them), but even if they don't you should put them in try-finally blocks to ensure they get closed no matter what happens in the runtime.
- I recommend using a linter like flake8 to find things like multiple or redundant imports (e.g. you import numpy three times in dh2l_db.py). It will also help you adhere more closely to the PEP8 standard, which will make your code more readable and reusable.

Overall, the code has a non-Python feel to it (I don't know Fortran, but I've read a lot of code from geophysicists and they often write code like this!). Patterns like functions mutating global variables are not common in Python. Typically, one would pass the variable in to the function, then return them to the user (these are so-called 'pure functions'). You will find this easier to maintain. It feels strange to me to run a function like dh2l.litho_dico(litho_dic_file) and not have something (like a new DataFrame) returned to me.

In summary I think dh2loop is an interesting and useful project that might help others pre-process data for machine learning and other kinds of modeling. I believe a shorter paper will have more impact. And there are some licensing questionmarks that need addressing.

I hope you are able to push this work forward and wish you the best of luck with the tool.