Geosci. Model Dev. Discuss.,
https://doi.org/10.5194/gmd-2018-180-RC1, 2018

Geoscientific Model Development Discussions

Interactive comment on "TPVTrack v1.0: A watershed segmentation and overlap correspondence method for tracking tropopause polar vortices" *by* Nicholas Szapiro and Steven Cavallo

B. Langenbrunner (Referee)

blangenb@uci.edu

Received and published: 23 September 2018

Review of "TPVTrack v1.0: A watershed segmentation and overlap correspondence method for tracking tropopause polar vortices," by Nicholas Szapiro and Steven Cavallo

This paper presents a new method and accompanying Python package for tracking tropopause polar vortices/TPVs. Generally, I find the paper very clear and well written, and as a non-expert on TPVs, I was able to follow it quite easily. I also think the authors did a nice job of differentiating their work from that of previous studies.

Because the subject matter is far from my area of expertise, I'm not able to comment on how this method compares to the others discussed in the paper. Instead, I was asked to review the Python-related aspects of this manuscript, and my comments and suggestions below are related to this.

I understand that a large amount of work goes into making a package like this portable, and my experience cloning the GitHub repository and changing parameters to run the "demo" case was fairly easy, though I did spend some time troubleshooting, and I've provided some more text to make proper adjustments below. Overall, I think this paper reads well and that the accompanying GitHub repository, with some minor changes, is suitable for publication in GMD. I don't think the major suggestions ("BROAD/MAJOR COMMENTS") need to be addressed for publication; I intend for them to be constructive comments for the authors to consider down the line.

GENERAL QUESTIONS/COMMENTS

This code repository seems very well written and logically structured, though a general comment is that while looking through the .py scripts, I desired a bit more documentation throughout. For example, a docstring below each function's definition (def . . .), briefly stating what it does (and/or its input/output) would be helpful in troubleshooting and reading the source code. See https://www.python.org/dev/peps/pep-0257/.

I only tried the simplest demo case with the provided ERA-Interim data set, and I haven't experimented with other data or running in parallel, so this review concerns the "unified" branch on GitHub.

BROAD/MAJOR COMMENTS (FOR FUTURE VERSIONS OF CODE)

I don't think this is important for the current manuscript, but I imagine it will eventually be useful to migrate this package to Python 3.x. I know Python 2.7 has a large user base, but using 3.x would likely give it more portability and would make it easier to use with other packages that are being widely adopted in the NetCDF community like

xarray and dask (for parallel computing).

In the same vein, I think it will also be good to migrate the plotting aspects over to cartopy eventually, since basemap has an end-of-life date sometime in 2020 (see the announcement here: https://matplotlib.org/basemap/users/intro.html#cartopy-new-management-and-eol-announcement). Again, not important for the current manuscript, but something to consider for future versions.

tpvTrack works great as a collection of scripts, but it's not precisely at the level or structure of a Python package. I take it that this is not the authors' intention for this initial publication, but if that is an eventual goal, some restructuring would probably need to take place. There's a nice tutorial for this here: https://packaging.python.org/tutorials/packaging-projects/.

The documentation is well done and easy to read, though I can see that it's a work in progress. If the authors hope that this package grows and is adopted by a broader user base, they might consider creating a documentation on Read the Docs using sphinx. This is not necessary for the current version, but again, something to consider for future versions and maintenance.

SUGGESTED MINOR CHANGES TO DOCUMENTATION

A few other things that would help clarify how the package works within the documentation would be:

* A clear list of output from each of the different options at the bottom of driver.py. I used demo() and demo_algo_plots(), but I didn't experiment much with the others. Also, adding some language in the documentation about the plots that are created and the .nc files that are produced would make them more user friendly. I wasn't sure what "corr_debug_X.png" and "seg_X_...png" corresponded to at first.

* A sentence somewhere stating that .pyc files will be created (but ignored in .gitignore) while running driver.py would be useful.

* The documentation only lists the main modules, but I would recommend adding the minor ones, as well, to be comprehensive.

* When running the demo, the "info" variable should also be changed so that global .nc attributes are appropriate, so this ought to be added in section 5.

SUGGESTED CHANGES TO CODE/REPOSITORY

The changes below helped me to successfully run the demo. It's totally up to the authors whether they want to implement all of these changes.

* I cloned and forked the GitHub repository and ran into a few issues with dependent package versions. To make the Anaconda environment easy to reconstruct, I would therefore recommend that the authors provide an environment (.yml) file that details the package versions that are installed. Then, Anaconda users could quickly install a new environment and (ideally) be up and running more quickly. For example, basemap doesn't work when installed using its default channel (there's a matplotlib issue with "set_axis_bgcolor()", which is a deprecated function). Instead, I needed to download it from conda-forge. The authors can create an environment file quickly using this tutorial: https://conda.io/docs/user-guide/tasks/manage-environments.html#sharing-an-environment.

* It might be a good idea to add a test-tpvTrack directory explicitly to the repository, so that users can work with the package out of the box using the default ERA-Interim data set. For example, as a user, I would ideally like to download the package and see a filled test-tpvTrack directory with successful output. The pull request I've created has these additions. This makes the GitHub repo slightly larger ($\sim$100 MB now, given the additional .nc files that driver.py creates), but I think it's still a reasonable size and well within the hosting limits and allows users to see what a "successful" run should look like.

* In changing the file variables for the package to properly run the demo, I noticed the

documentation says to modify things like fDirData in preProcess.py, but I had to modify these variables in my_settings.py. I ended up modifying my_settings.py and would recommend that the authors change section 5 in the documentation to reflect this. I also changed the directory variables (like fDirData) to use relative links rather than full links, which may help users run the demo code with fewer modifications.

* I changed the color map in the segment.py script to use the RdBu_r color bar, so that it looks closer to Fig. 1 of the manuscript.

Interactive comment on Geosci. Model Dev. Discuss., https://doi.org/10.5194/gmd-2018-180, 2018.