Criteria	Description of how we meet criteria	Location of relevant files (if applicable)
Documentation	We wrote detailed functional and design specifications, made a README.md in the top-level directory, made READMEs with additional detail in some subdirectories, and provided abbreviated documentation as a part of our dashboard itself.	Documentation in `docs/`; README files throughout the repository
Organization and project structure	The project structure is organized as specified in the project requirement with `local_climate_change_tool` module and `phase1_data_wrangler` and `phase2_dashboard_generator` submodules containing `__init__.py` files. The project contains all required files (README.md, LICENSE, `setup.py`), a docs folder, an examples folder, and properly named python package folder and test scripts.	The structure of our repository is summarized in the README in the top-level directory
Data sources	We used two data sources: (1) climate model output of monthly mean temperature and (2) historical observations of monthly mean temperature. The climate model data itself also consisted of several data sources (i.e. model output from different climate models).	Description of data sources: here (`docs/Data_Description.pdf`)
Code quality	All code scored 8 or higher in pylint as demonstrated in screenshots of pylint scores. Overall pylint score was 8.73.	Screenshots of pylint scores are in `docs/pylint_scores/`
Test coverage	Our contains 5 unit tests (each containing multiple test functions), for all 6 python scripts in the `phase1_data_wrangler` module of our package. This is 82% test coverage	Screenshots of running `pytest --cov` and additional documentation of our testing process are in

	based on running `pytest --cov` as described in the README in the testing docs directory. We did not write tests for the dashboard generation component of our code based on input from Dave on 11/18/2019 that this was "beyond the scope of this class" and not necessary for our final project.	`docs/testing_docs/`
Example quality	We wrote a detailed user guide with examples of how to launch and interact with the dashboard generator, including screenshots of how to use the dashboard.	User guide in `examples/`
Continuous integration	We implemented continuous integration via travis-CI. We achieved a successful build (as documented by the flag in our Github repository).	Travis-CI for project here
Completeness of setup.py script	The `setup.py` script contains metadata about the package and elements to initialize the project, including requirements, repository information, and a description. Additional requirements are installed by the user via the file `requirements.txt` as explained in the README and User Guide.	`setup.py` in home directory
Creativity and technical challenge	The datasets presented a significant challenge since each model had a different grid and time variable and had to be processed to be consistent with each other and with the observations. The data was too large to upload to GitHub, so extra effort was made to accomodate downloading the data for the user. The design of the panel makes it easy for a user to create and view plots of climate data, which are frequently used by scientists and in major climate reports.	Motivation for project described in final presentation, which is saved in: `docs/Final_Presentation_Dec_4.pdf` Detailed documentation of technical challenges and lessons learned saved below

Lessons learned:

- How to deal with merge conflicts
- Establish a consistent environment for development (across different team members and computing platforms) early on (e.g. we were having issues because of Python 3.7 vs. 3.8, but did not recognize this issue until the last week of the project)
- Write unit tests as you develop code, not after (they can help identify bugs!)
- Comment code while you go so your comments are fresh and more accurate instead of waiting and possibly forgetting details/exceptions or forgetting to add them at all
- Be conservative in defining a project's scope given the available time. We defined a very lofty scope, and ultimately had to drop several features we had initially intended to incorporate. Additionally, even though we had a working data wrangler and dashboard generator two weeks ago, we still needed at least two weeks to address environment compatibility issues, improving test coverage, and make comprehensive documentation and a user guide.
- Projects that hinge on use of very large datasets pose challenges for continuous integration and use on personal computers.