Quality document Luminous pollution analysis

Damien DASSEUX - Etienne MALACARNE - Andrew MARY HUET DE BAROCHEZ- Maxime PAULIN - Valentin RICHARD - Jordan PRUVOST

1. Python

For this project we will use the version 3.10.9 of python with the official venv module.

1.1. Virtual environment

To install the python environment you must ensure to have a version of python 3.10. Run the following command to create a virtual environment to the desired path.

```
python3.10 -m venv /path/to/venv
```

You can then launch the environment with the command source followed by the path of the active file in the venv.

```
source /path/to/venv/bin/active
```

1.2. Dependencies

In the venv, make sure to install python dependencies before running the project.

```
pip install -r requirements.txt
```

To update dependencies you can run this command:

```
path/to/venv/bin/python -m pip freeze > requirements.txt
```

2. Linter

The linter helps us keeping the code clean, it should idealy be run and the code cleaned before every commit.

If pylint wasn't installed with the requirements you can do it by hand.

```
pip install pylint
```

If you want to integrate pylint to pycharm:

https://stackoverflow.com/questions/38134086/how-to-run-pylint-with-pycharm

Run the linter with the following command:

```
pylint [module] or [file] or [directory]
```

https://pylint.pycqa.org/en/latest/user_guide/usage/run.html

3. Code style

Code style of this project will follow PEP8 rules: "https://peps.python.org/pep-0008/"

Here are some main code style

- Use 4 spaces per indentation level.
- Limit all lines to a maximum of 79 characters.
- Functions, parameters and variable names should be written in lower case.

- Try to use type hints as much as you can.

Please lint your code before committing anything.

4. Source version control

We use github as a source version control. Here is the link of the project: https://github.com/Onyx39/luminous-pollution-analysis

4.1. Commit message

The team name (e.g what main feature one is working on) should be inside brackets. Then a small message should explain what changes / fixes have been proposed.

Each commit should have one purpose, and should be thoroughly explained if otherwise.

4.2. Branches

Everything in the main branch should be working, without smell code and errors. To push code in main you have to create a pull request from another branch.

4.3. Pull request

When a branch reaches a level of stability and quality so that it can be merged, pull request is done. At this point, a code review is done by a peer (ie anyone else that didn't participate in that code) is performed. If the code conforms to the reviewer's expectation, it is merged. Otherwise, the code shall receive minor updates, or be declined, and it will be the initial developer's duty to make it conform to those expectation

5. Testing

Once the codebase will be solid enough, we will add unit tests. These tests will be written by another party that the one that originally wrote the tested code. Human resources

6. Human resources

People	Responsibilities
DASSEUX Damien	• Fetching sentinelle II data with Copernicus API / Sentinelle hub • Computing NDVI over time • Computing distance between cities and forests
RICHARD Valentin	• Extraction of the forest data and processing • Displaying forest boundaries on a map • Displaying cities on a map • Refactoring code
PRUVOST Jordan	• Extraction of the forest data and processing • Gathering cities data • Display cities on the map
PAULIN Maxime	• Helping fetching sentinelle II data with Copernicus API • Establishing a github policies
MALACARNE Etienne	• Exploration of sentinelle II API • Computing NDVI values • Computing of the luminance

	• Gathering cities data
MARY HUET DE BAROCHEZ Andrew	• Ensuring the quality of the project • Establishing a github policy • Refactoring and improving code readability • Contributed to documentation • Used shapely to reduce the number of points in forest boundaries