Data visualization of O3 levels in Europe

5011CEM
Big Data Programming Project Module

24th of April of 2020

1. Introduction

The visualisation of O3 levels in Europe was created to satisfy client requirements. For that, client instructions were followed and applied to extract large amounts of data and plot it in a simple and effective manner. The software uses python in conjunction with matplotlib, cartopy and some other modules to plot the data in a map and create the images requested. A label to better understand the relation between the representation and the data values was solicited.

With this, twenty-four images were created, where each demonstrates the O3 levels analysed using different models, eight in total. A centralized map in Europe was implemented using the extracted coordinates of Europe from the data file, along with a side-by side-disposition of the graphs for a better visual comparison. In addition, a colorbar as a visual label was displayed at the bottom of each image. The software creates a default directory named O3_images where it saves all twenty-four images named with their respective time and png extension. This directory is created in the same directory the program is executed.

2. Specification

The integration of SMART targets is essential for the delivery of a quality product. The targets ensure the client requirements are meet and there are not delays with delivery. The projects respect the following targets:

- Data is collected from the client in the first week of the project development. This is presented in a nc file.
- Analysis and understanding of the project, subproject and the data given for a better manipulation of the data, which consequently allows for a better efficiency when codding the software. The IDE is chosen based on the projects requirements and file provided. Python was chosen to develop the project, which consequently requires several modules to support this data extension, extract and plot the data, such:
    i. Numpy
    ii. Xarray
    iii. Matplotlib
    iv. Cartopy
    v. Concurrent.futures

- After the IDE is chosen, data is required to be manipulated and sliced with numpy due to it being constrained in a 4D matrix format. The necessary variables are then extracted to be used for the plotting and visualization of the data in a heatmap.
- The data plots the 24 images with 8 maps in each based on the coordinates included in the 3D matrix of each model.
- Finally, the images are saved in a directory created by the software. In the same destination of the software. Furthermore, testing and error handling will be done to ensure the product quality and requirements.

Limitations

The software runs in every machine. However, some might take longer periods of time to process the graph than others. A way of going around this issue was to provides the choice of opening the graphs or just save them in the directory. Although, the saving process also has a waiting time of three or four minutes depending on the machine.

3. Code

The code is divided in 4 main parts. The first detects the only file in the directory with a nc extension and open the data file using xarray. The coordinates, time and models are extracted using uring numpy, which converts the variables into numpy arrays. This facilitates the management of matrixes, which will be essential for the data plotting.

The second part is the user input, where the user can choose between five colour pallets to visualise the graphs with based on what fits better for the user's needs. This provides some accessibility for colour-blindness. It also asks if the graphs should be plotted or just saved.

Third, plotting of the graphs. This begins by creating the window and adding the windows title which depend on the hour that is represented by that window. Then it loops through eight models, where five rows and two columns of graphs are created. The last row is left empty. The map is then drawn, the axis created, the name of the model added to the graph and the is data plotted for each model in that specific hour. To plot the data of that hour, the numpy array is required to be sliced since each model consists of a 4D matrix. The slicing process is essential to extract each layer and obtain a matrix with the O3 values for a set of longitude and latitude. Later, once the loops are finished, the colorbar is plotted and its measurements as well. Lastly, the code to save the windows in png in a created directory named O3_images and "quit all" button, which sets the close all windows event in a button.

Finally, the fourth part. This part is shortest in terms of code, but one of the most essential ones. It is responsible for the parallel processing and it calls the function to plot the graphs, while passing the hours as parameters via a loop, which creates twenty-four process at the same time. The parallel programming allows the code to plot the twenty-four windows synchronously, reducing the execution time from twenty minutes to six minutes without showing the graphs. This depends on the machine used.

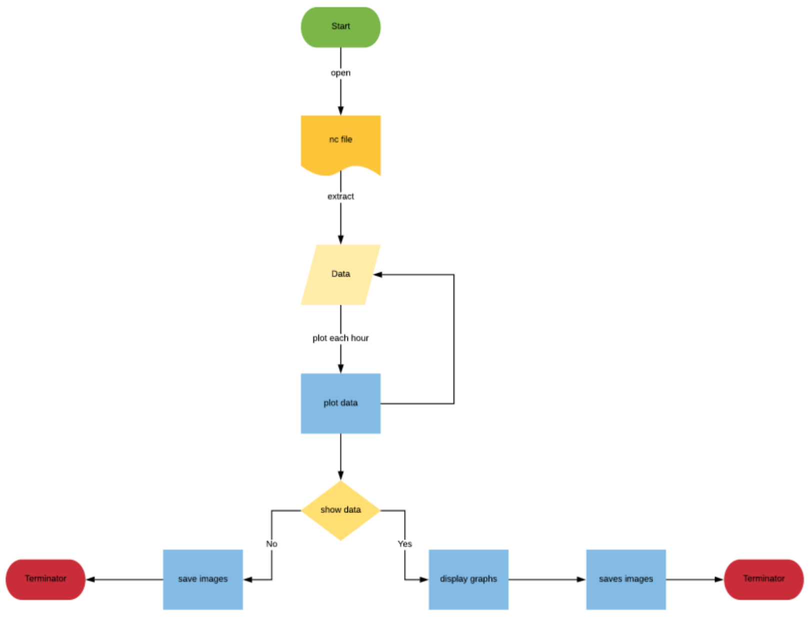

Figure 1 – flowchart of visualization of O3 levels in Europe

Outstanding Issues

One of the aspects for an efficient and complement software is the automated tests using TDD. This was not implemented as TDD and not all aspects of the code were tested.

4. Results & Conclusions

This project outputs twenty-four images saved in a directory called O3_images, which is created by the software. Each image plots 8 graphs representing each model per hour. The images are saved with a png extension.

Visualization is a heavy weighted process, which takes time even with parallel processing. The plotting of the graphs as required by the client was successful. However, when images are saved, they lose quality, displaying some level of pixelation. As requested by the client, a few colour sets were included for accessibility purposes. Despite, the available colours might not be ideal for colour-blindness as some colour sets have similar colour contrast levels.

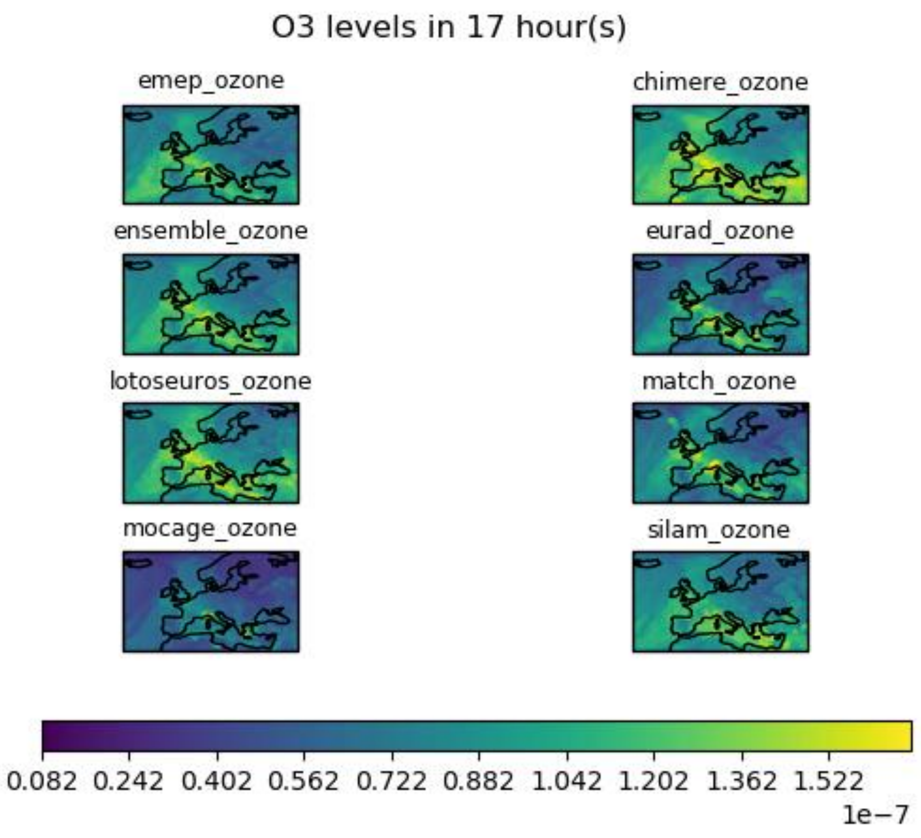

Figure 2 - output image from software

5. Future Work

In the future, for a better performance, MATLAB could be experimented with to come to a better conclusion of which software provides a better execution of the code and more image quality. In addition, research for colour-blindness can be carried out for better decision in selecting colour sets for the user.

6. Summary

This project satisfies all client expectations and requirements. From the plotting of the graphs, to the selection of colours to plot the graphs with. The graphs labels were also implemented in an easy to understand manner and the effectiveness of the execution time implemented as well through parallel processing. However, the testing processes still requires some attention, having a lot of issues still.

Appendices

Specification Document

1. Introduction

This project allows the visualization of O3 levels in Europe by comparing all models side by side in each correspondent hour. This is essential as it's a for of communications and allows the analysis and understanding of the data.

2. Project Requirements

The software overlays a map of Europe and plots the intensity of the O3 in a specific location. The different levels of O3 intensity are represented with a color scheme. In addition, a color bar is included for an easy understanding of the correlation of the O3 levels and its color representation. The user can also change the colors displayed from a list.

2.1 Related documents

Component	Name (with link to the document)	Description
Using Cartopy with matplotlib	Cartopy Documentation https://scitools.org.uk/cartopy/docs/latest/matplotlib/intro.html	Instructions and tutorials on how to use Cartopy with matplotlib
Executor objects ProcessPollExecuter	Concurrent.futures Documentation https://docs.python.org/3/library/concurrent.futures.html	Instructions and tutorials on how to parallel process

2.2 Terms/Acronyms and Definitions

Term/Acronym	Definition	Description
O3	Ozone is an unstable and poisonous gas that is formed naturally from oxygen.	Inorganic molecule, that is formed of 3 molecules of oxygen. The gas formed has a pale blue color and a pungent smell.

3. Risks and Assumptions

In this project the data of the models is assumed to be in .nc format and the .nc file to be in the same folder as the python program code, bigdata.py. The models are expected to be the following: emep_ozone, chimere_ozone, ensemble_ozone, eurad_ozone, lotoseuros_ozone, match_ozone, mocage_ozone and silam_ozone. If models are different, the code will require to be adjusted.

This project runs with cartopy, which can be installed with pip. However, an issue with the installation arouse and was then installed through Anaconda. For the requirements.txt to be successfully it is assumed anaconda is installed through the terminal.

In this project, there is a low risk of failure, how if the inputs are not in the correct format the software will not run. When showing the graphs, there is a risk of slower than normal graphical processing if the machine used to run it is slow. Visualizing eight graphs per window and a windows per hour is a heavy process, which might case the "Quit all" button to take extended amounts of time to perform the event. Yet, this issue does not affect the overall performance of the software.

4. Out of Scope

In this project no mathematical comparisons of the models will be calculated, only visual comparisons in map form. The GBE values will not be used in the visualization and lastly, the animation of the graphs (gifs) will not be available.

5. System/ Solution Overview

The software plots the O3 levels in an overlaid map of Europe to compare the eight data models in a visual manner. Each image represents the O3 levels in a specific hour, creating 24 images saved in O3_images folder that is created by the software in the same folder as the python program.

5.1 Context Diagram/ Interface Diagram/ Data Flow Diagram, Application Screen Flow, Sitemap, Process Flow

This project will have a very simple interface. The first user interaction will be terminal based to choose the colour of the representation of O3 levels. The user will be able to choose between 5 different colours. After the colour is chosen, the graphs will be processed. Twenty-four maps will be only saved or saved and showed to the user. Each window will have eight graphical representations, one of each model. Additionally, a quit button to close all windows at once.

Terminal based interface:

```
File <filename.nc> is being uploaded
Coloursets:
    1: colour1
    2: colour2
    3: colour3
    4: colour4
    5: colour5
Please choose a colour:
Do you want to show the graphs? (yes/no)
```

Graphical user interface:

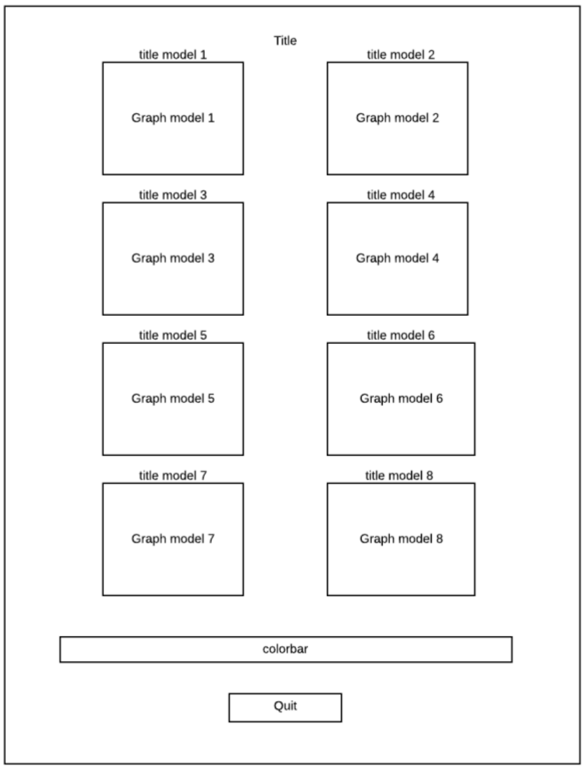

Figure 3 - user interface schematics

6. Project Management

The project will be divided into eight parts, each part will consist of three or four tasks, which will be completed in a week's time. The first 4 weeks will focus on understanding the data and the required data models as well as creating a data flow graph and interface. A Kanban board will be used to track each task through the weeks from uncompleted to completed. A Github repository will be used to version control the code every week after each task. After the code is completed, the weeks after until the submission dates will be focus on preparing the portfolio, the report and the viva.

7. References

https://www.visual-paradigm.com/guide/data-flow-diagram/what-is-data-flow-diagram/

8. Open Issues

Issue ID	Issue	Raised By	Raised On	Solution/ Decision	Resolved By	Resolved On	Status
1	Geos raising error in terminal	Team member 1	bigdata.py	Change to cartopy	Team member 1	bigdata.py	closed
2	Pip gives error when installing cartopy	Team member 1	bigdata.py	Install anaconda	Team member 1	bigdata.py	closed
3	Plotting process is slow	Team member 1	bigdata.py	Parallel processing with concurrant.futures	Team member 1	bigdata.py parallel_processing()	closed
4	Showing graphs is slow and takes a long time	Team member 1	bigdata.py	Save graphs in directory instead of showing them	Team member 1	bigdata.py make_graph() line 102	closed
5	Quit all button not working	Team member 1	bigdata.py	Had to be moved inside of make_graph()	Team member 1	bigdata.py make_graph() Line 126	closed