Independent Project: Water Quality Data Downloader/Analyser/Viewer

WORD COUNT: **1021**

1 Introduction

The Environment Agency (EA) hosts a Water Quality Archive (WQA) online[1] (Figure 1), providing public access to sampling data from 58000 water quality sampling points across England under the Open Government License. Therefore, the archive is a significant source of water quality data which could prove very useful in further academic research and industry projects.

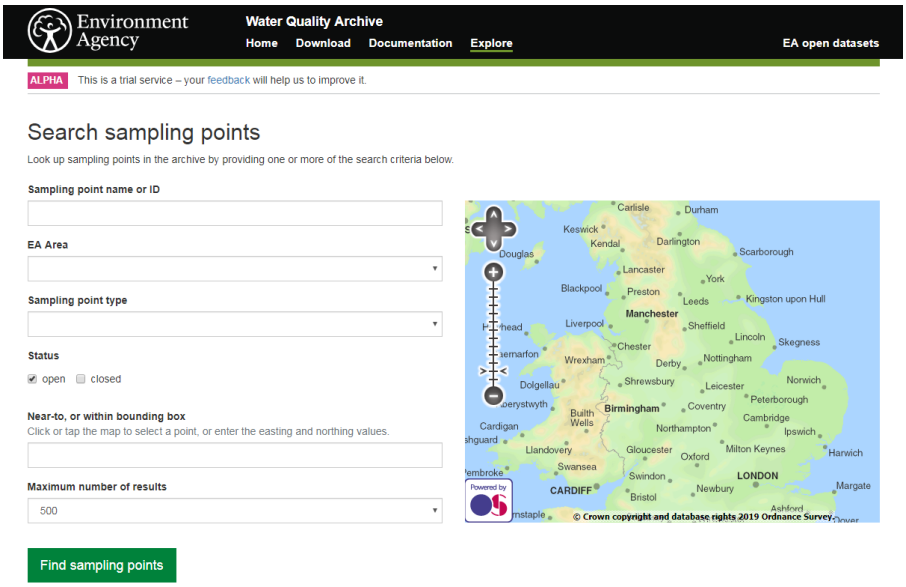


Figure 1: Screenshot of Environment Agency Water Quality Archive website.

The EA currently offers three methods of obtaining data from the WQA:

1. Download pre-defined datasets for a given year and location from the website.
2. Use the online Explorer tool to search by sampling point name/ID, by EA operational area or by a bounding box.
3. Access via an Application Programming Interface (API).

Whilst each of these methods are useful, they all offer their own drawbacks. For example, the first method only allows users to download very large datasets, the second will only allow the user to search in specific areas, and the third method requires the user to have significant coding knowledge.

Therefore, there is currently a gap in the methods available for downloading the water quality data. As such, the purpose of this project is as follows:

[1] https://environment.data.gov.uk/water-quality/view/landing

Aim: To develop a GIS-based tool (written in Python) to download, analyse and view water quality data from a given area of interest which may be used by individuals with limited/no coding skills and GIS knowledge.

Objectives:

1. To apply Python methods and modules exercised during GEOG5790 practicals.
2. To practice data manipulation using packages such as pandas and csv.
3. To utilise ArcGIS toolboxes and the arcpy module to undertake geoprocessing tasks.
4. To develop an interactive dashboard for graphing, mapping and tabulating data using Jupyter Notebook, widgets and plotly.

2 Code

2.1 Structure

As illustrated in Figure 2, the code is split into 3 stages.

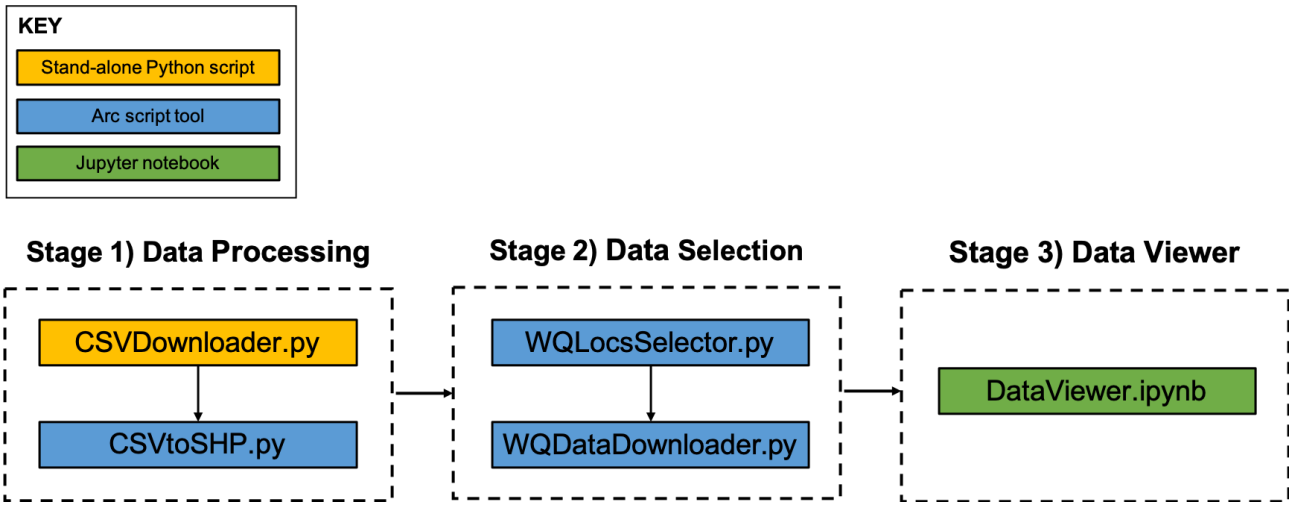


Figure 2: Schematic to illustrate code structure.

2.2 Use

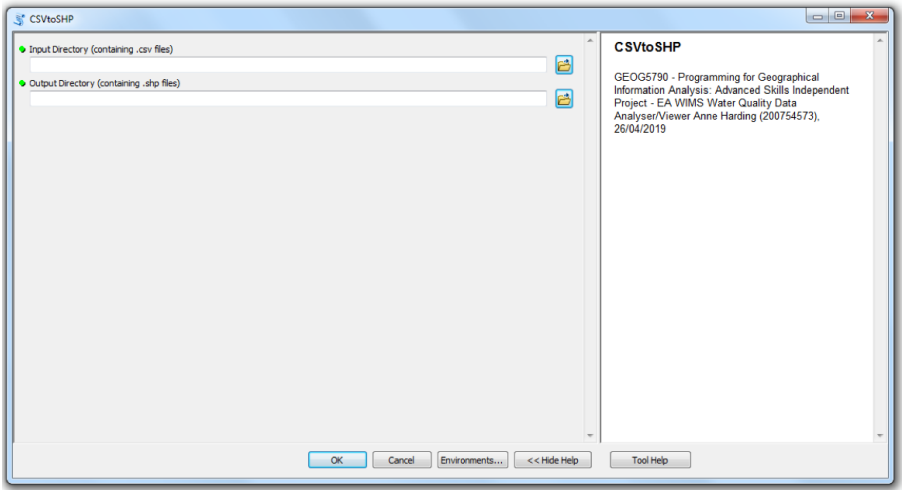
2.2.1 Stage 1) Data Processing

Stage 1) CSVDownloader.py script downloads all data from the EA WQA and concatenates all years of data for each EA operational area into individual .csv files. The CSVtoSHP.py Arc script tool (Figure 3a) creates a .shp file containing all EA water quality monitoring locations in England.

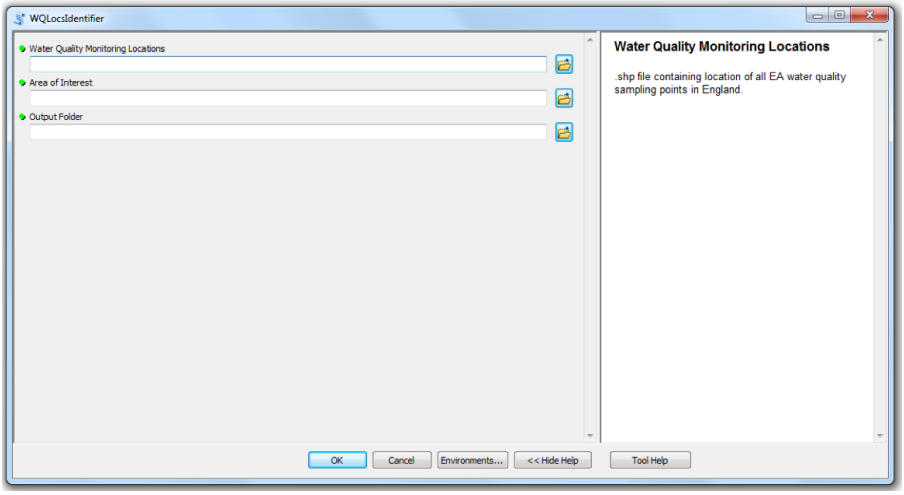
2.2.2 Stage 2) Data Selection

Stage 2) WQLocsIdentifier.py Arc script tool (Figure 3b) identifies the water quality sampling sites within a user-specified area of interest. Then the WQDataExtractor.py Arc script tool

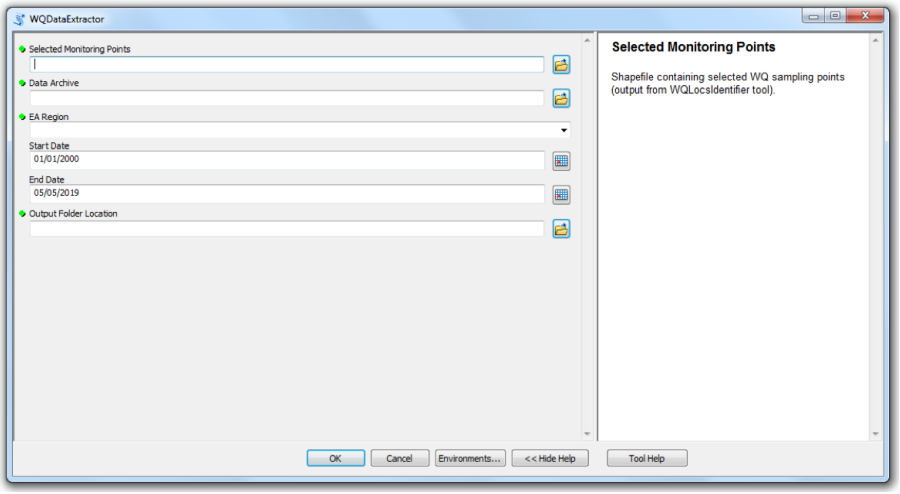
(Figure 3c) allows extraction of data for the selected sites within a user-specified date period, outputting a .csv file containing the extracted data.



(a) Stage 1) CSVtoSHP.py.



(b) Stage 2) WQLocsIdentifier.py.



(c) Stage 2) WQDataExtractor.py.

Figure 3: Screenshots of user interfaces for Script Tools contained within ArcGIS Toolbox (WQToolbox.tbx).

2.2.3 Stage 3) Data Viewer

Stage 3 allows the user to graph, map and tabulate the .csv file from Stage 2) containing extracted data in Jupyter Notebook using widgets to filter the dataset by determinand (Figure 4). The filtered dataset is then graphed using plotly to produce an interactive plot, with the locations of each of the sampling points plotted up on a map using folium. Descriptive statistics for the filtered dataset are calculated using the pandas describe() function. The graph and map are saved as .html files, with the statistics table outputted as a .csv file.

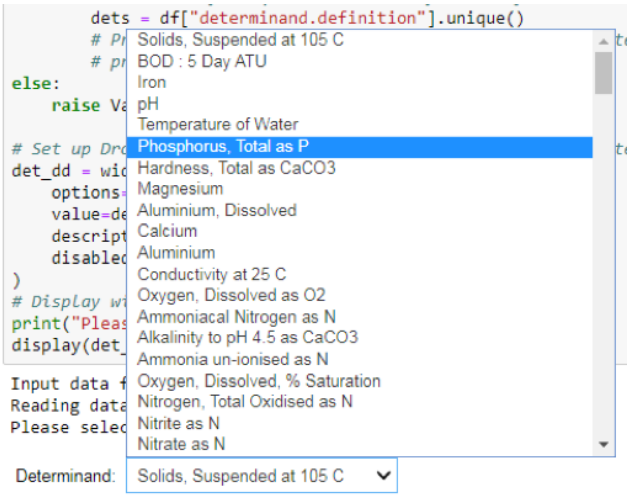
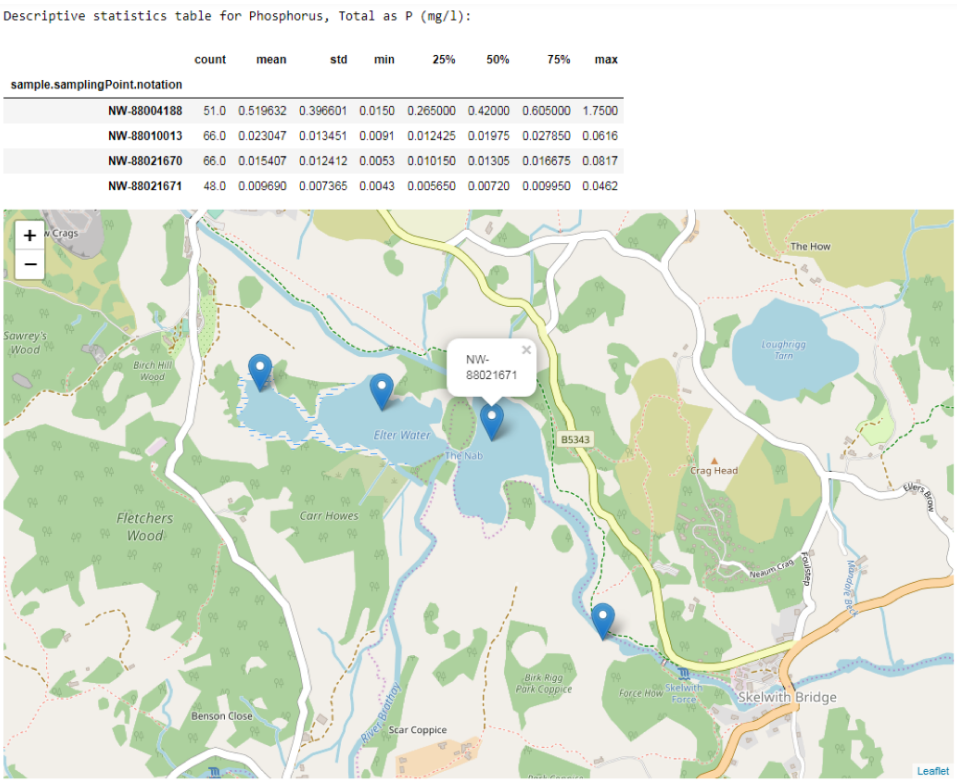


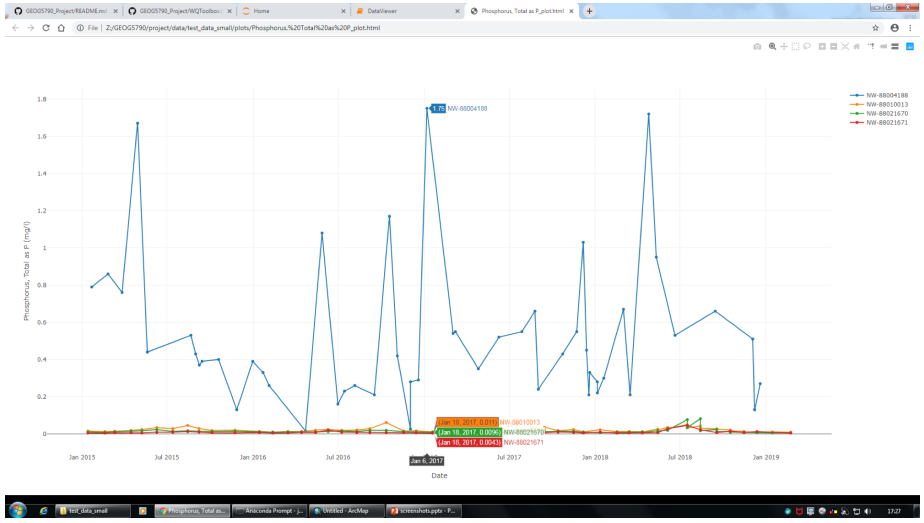
Figure 4: Screenshot of dropdown widget in DataViewer.ipynb for user to select determinand.



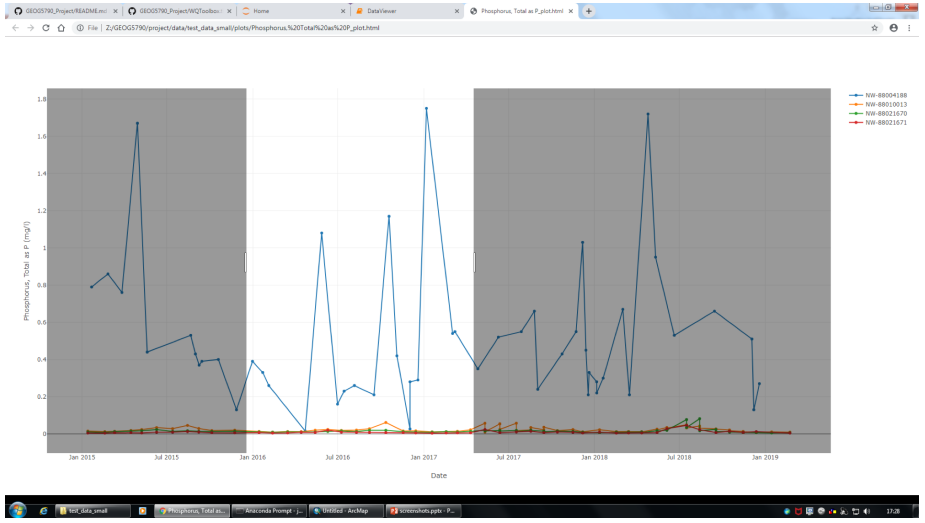
Descriptive statistics table for Phosphorus, Total as P (mg/l):

sample.samplingPoint.notation	count	mean	std	min	25%	50%	75%	max
NW-88004188	51.0	0.519632	0.396601	0.0150	0.265000	0.42000	0.605000	1.7500
NW-88010013	66.0	0.023047	0.013451	0.0091	0.012425	0.01975	0.027850	0.0616
NW-88021670	66.0	0.015407	0.012412	0.0053	0.010150	0.01305	0.016675	0.0817
NW-88021671	48.0	0.009690	0.007365	0.0043	0.005650	0.00720	0.009950	0.0462

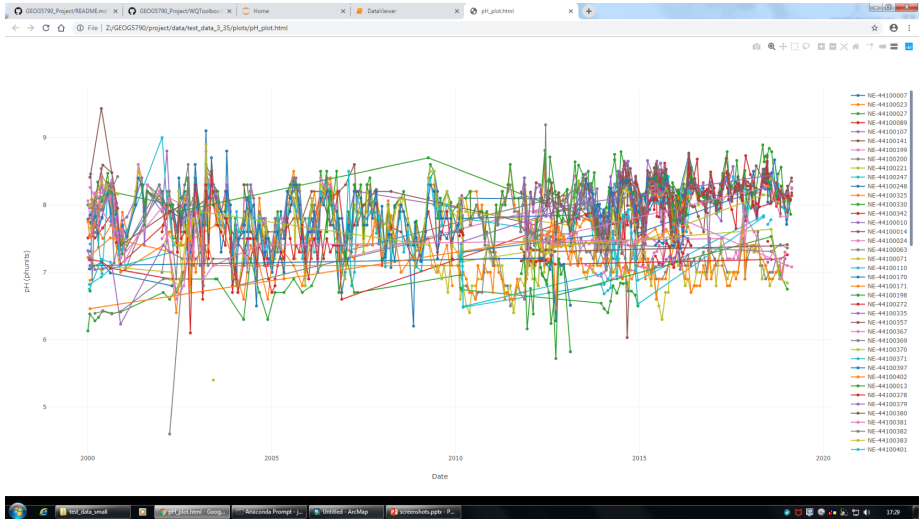
Figure 5: Screenshot of descriptive statistics table and map in DataViewer.ipynb, including pop-up text for each point showing sampling point label.



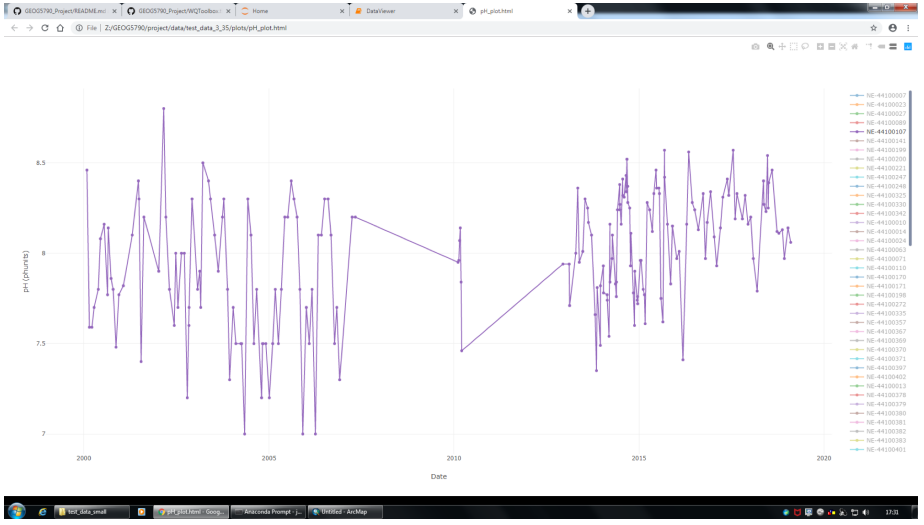
(a) Plot with small number of traces, showing data point information.



(b) Interactive time filtering of graph using plotly functionality.



(c) Plot with large number of traces (>50).



(d) Interactive filtering of legend items using plotly functionality.

Figure 6: Example graphical outputs from DataViewer.ipynb.

2.3 Runtime

Stage 1 takes a relatively period of time to run (2-3 hours), but only needs to be run when the WQA is updated (every few months). It is envisioned that once the data has been downloaded and processed using Stage 1, Stages 2 and 3 can be run whenever the user wants to extract and analyse the water quality data for a given area of interest. Stage 2 takes approx. 2 minutes to run, and Stage 3 takes less than 1 minute to run.

2.4 Development

During the early stages of development, I tried to make use of the EA WQA API to extract the data from the website using the requests package to extract the data in .html format, however I found the API very difficult to work with and resorted to downloading the data in .csv format.

I orginally intended for the functionality of the two Stage 2 Arc script tools to be contained within one script tool. However, I found that Arc automatically applied a lock to the datasets in order to prevent issues with multiple editors and overwriting of data such that it was impossible to work with the 'wqPointsclip' dataset once it had been created. Therefore, I had to split the functionality into two scripts in order to allow the geoprocessing tasks to be undertaken. This means that there are more tools for the user to run and reduces the efficiency of the project, but there were no other workarounds available.

I tested Stages 2 and 3 of the tool using two example datasets; one smaller (26 sampling points) and one larger (116 sampling points). The larger dataset has very limited effect on the runtime of Stages 2 and 3. However, Figure 6c shows that there can be too many traces on the graph in order to be able to visualise the data properly. In this case, the interactive plotting functionality of the plotly graph may be used to filter the number of traces on the graph (Figure 6d) or to restrict the time range visible on the graph (Figure 6b).

3 Summary

This project has created a tool which allows a user with limited coding and GIS knowledge to extract, plot and analyse water quality data for a given area of interest. Previously this may have been undertaken by downloading data manually and analysing the data using software such as Excel. This method is prone to human error and may result in problems with the data outputs and subsequent analysis. Therefore, this tool provides a consistent method for water quality data analysis, reducing the risk of human error significantly. Additionally, the tool standardises the format of the outputs of such analysis (map, graph and statistics table).

Future development of the tool might include setting up a task scheduler to run the Stage 1 CSVDownloader.py script at regular intervals (i.e. every 3 months) in order to ensure that the WQA is always up to date. Similarly, to improve the usabiity of the Jupyter Notebook data viewer in further development, I would write the majority of the code in functions saved in a separate script which may be called by the .ipynb file. This would reduce the number of lines that the user sees, and would also reduce the potential for human error in accidently altering lines of code.