|  |  |  |  |
| --- | --- | --- | --- |
|  | | logo_black.gif | |
|  | 940065422 |  |  |
|  | GEOG5995M: Programming for the Social Sciences – Core Skills |  |  |
|  | Assessment 2 – The Black Death |
|  |  |

|  |
| --- |
|  |
| 1. |
| 2. |
| 3. |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

**GEOG5995M: Programming for the Social Sciences – Core Skills**

**Assignment 2 – The Black Death**

**Intention of the software**

The objectives for this assignment were to:

* Read in some data
* Process it in some way
* Display the results
* Write the results to a file

I chose to use one of the given projects – The Black Death (Evans, 2017). The aim of this project is to display the data sets, calculate the average and total deaths per week from the Black Death in 1665 for 16 parishes using a given equation and enable the user to change the parameters of the equation to examine the impact on the deaths. The guidance that I followed is here:

<http://www.geog.leeds.ac.uk/courses/computing/study/core-python-phd/assessment2/death.html>

Using this guidance I developed a Jupyter Notebook that performs the following steps:

1. *Reading in the two datasets*

The first dataset is a raster map of the average number of rats caught per week by rat catchers per 100m x 100m square (parameter ‘r’). The second is a raster map of the average population density per 100m x 100m square for 16 square parishes (parameter ‘p’).

1. *Processing and displaying the data*

This step involves creating images of both datasets that are appropriately labelled with colour schemes that make sense for the type of data they are displaying.

1. *Calculating and displaying the average number of deaths per week*

The average number of deaths per week (‘d’) is calculated using the equation given by Evans (2017):

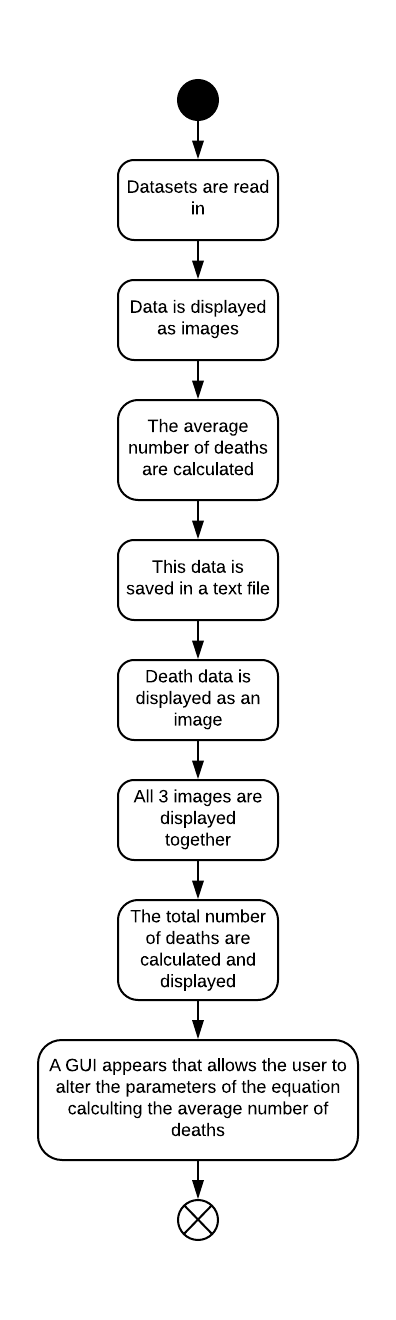
This output is displayed as another map.

1. *Displaying the three maps together*
2. *Saving the average deaths map as a text file with each line in the file equal to a line on the map*
3. *Calculating and displaying the total deaths per week*
4. *Allowing the user to change the parameter weights for the equation*

The Jupyter notebook and original Python code can be accessed here:

<https://publichealthdatageek.github.io/research_page.html>

**UML diagram for the software**



**Development process and issues**

Overall this was a very challenging piece of work for me to deliver given that I had never coded anything before this module and had never used Anaconda, Python and Jupyter.

The first challenge was ensuring I was confident that I had imported and displayed the raster data appropriately. I utilised the information learnt from the module about raster data (each row is an increase in the y-direction and subsequent values in a row being an increase in the x-direction) and reading such data into a 2D array to inform my code. I cross-checked this by visualising the data in a spreadsheet and by checking the example images shown in Evans (2017).

Once I had imported the data, I then displayed it as two images. Initially these images had axis on that caused confusion and no indication of what the colours meant. Using Python documentation I discovered how to remove the default axis labels and add appropriate labelling and a colorbar. I tried different colormap schemes to identify the ones I felt conveyed the data well to the viewer. A same colour sequential colormap was used for population density whereas a different sequential colormap that enabled the viewer to clearly demark the different rat catcher patches was used for this data.

I then tried to calculate the number of average deaths. However, because of the way the data was structured I couldn’t do this. After reading some Python guidance, I realised that the easiest option was to convert the data into a numpy array. This enabled me to use floats for the equation calculation and to enable utilisation of the numpy functions for the calculations.

I then developed code to run the equation by creating new variables using *for* loops and the inbuilt behaviour of numpy arrays. One *for* loop looped through each item within each row (and then subsequent rows) multiplying each r value by 0.8 and the second multiplying each p value by 1.3. The numpy structure ensured that this worked correctly. I then calculated the average death for each 100 x 100 square by using a numpy multiply function. These average deaths per week were displayed as an image with appropriate labelling and a colorbar. I tested that these calculations were running correctly in numpy by performing test calculations (code not shown).

However, it is clear from the results that the equation given for this calculation that was developed from ‘small-scale studies’ is wildly inaccurate for calculating average deaths. As r values range from 0-192 and p values range from 0-255 then the average deaths (d), using the below equation, can theoretically be between 0 and 50918.4.

In reality, the range is from 0 to over 35000 but these are clearly inappropriate if the population density is in the hundreds. This suggests that the equation needs further development.

Going back to the assignment, the next step, which proved to be very tricky, was to create an image with all 3 maps together. Whilst it was relatively easy to get three images in the same horizontal plane and close enough together it proved impossible to get them to be the same size to enable direct comparison. I tried altering the saving of the 3 individual images (by specifying image size and dpi); used various matplot options such as figure, subplot, gridspec and close(); and searched StackOverflow and Python documentation but I was unable to get them all to be equal size. In end I settled for the code that seemed to give the ‘best’ result. For this I had to go back to the earlier code where the images were created and alter some of this (e.g. save the images, clear the figure) to enable the subsequent code to work correctly.

Trying to decide how to calculate the total number of deaths proved to be difficult. The guidance said that each raster data point was a value for a 100m by 100m square area. Therefore the sensible approach appeared to be to sum the average deaths for each 100m by 100m area to obtain the total deaths. I did this using the np.sum function and then applying a rounding process to give a whole number. Again I checked these calculations separately (code not included). However, this creates a very big number as it is based on the large numbers of average deaths and thus seems unreasonable. Another approach I considered was averaging the average number of deaths within each parish.

The final step was to create something that allowed the user to change the parameter weights for the equation. Having looked at various options (such as scrollbars etc), I decided to use PyQT5 to create a GUI to enable the equation parameters to be altered. I watched a series of YouTube videos created by Forogh (2018) on PyQT5 GUI programming to learn how to create a GUI. I also utilised another You Tube video to understand how to create the calculations and display the answers in the GUI (Programming Liftoff, 2017). The GUI programming approach was very much a ‘code and then run to see what happens’ and whether that corresponded to what I wanted. This part (the GUI) was extremely satisfying as it felt like the culmination of all my learning.

I then copied and structured the code appropriately into a Jupyter notebook. There appeared to be some initial issues with images and the GUI displaying differently but I think I resolved those.

**Sources of assistance**

The key sources of assistance were questions and answers in Stackoverflow (Stackoverflow, 2018) and the official Python and Matplotlib documentation (Matplotlib Development Tea, 2018 and Python Software Foundation, 2018). I also utilised YouTube videos (Forogh, 2018 and Programming Liftoff, 2017).

**References**

Evans, A. 2017. *Geography Programming Courses – The Black Death.* [Online]. [Accessed 5 December 2018]. Available at: <http://www.geog.leeds.ac.uk/courses/computing/study/core-python-phd/assessment2/death.html>

Forogh, P. 2018. *PyQT5 GUI Programming with Python 3.6*. [Online]. [Accessed 29 November 2018]. Available at:

<https://www.youtube.com/channel/UCD6ArU-AYbfIj5sx2L4SZAQ>

Matplotlib Development Tea. 2018. *Matplotlib*. [Online]. [Accessed 13 December 2018]. Available at: <https://matplotlib.org/>

Programming Liftoff. 2017. *PyQT5 Calculator Gui Tutorial Part 3.* [Online]. [Accessed 29 November 2018]. Available at:

<https://www.youtube.com/watch?v=urEz0Q7nXaU>

Python Software Foundation. 2018. *Python 3.80a0 documentation.* [Online]. [Accessed 13 December 2018]. Available at: <https://docs.python.org/3.8/>

Stackoverflow. 2018. *Stackoverflow – all questions.* [Online]. [Accessed 13 December 2018]. Available at: <https://stackoverflow.com/questions>