Black Death Python Project Reflection

Program intention

The aim of the program is to enable insights into the relationship between human and rat population density and numbers of deaths during the Black Death outbreak of 1665. The program is intended to support epidemiologists investigating the Black Death by presenting data visually as raster maps, as a total sum, and as a txt file.

The program reads in data of rat populations recorded by rat catchers and population density derived from parish records. The data is recorded according to 100m x 100m squares of London. The program displays this data as raster maps. The use of a greyscale colour scheme, with lighter shades indicating low density and darker shades indicating high density, enables intuitive, simple understanding of the data relationships.

Using the equation d = (0.8 x r) x (1.3 x p), the program calculates the estimated number of deaths for each square. The program displays this data as a raster map, displays the total sum of the estimated number of deaths, and saves the data as a txt file in the current working directory. The 0.8 and 1.3 weightings can be changed to different values through user input.

Potential uses for the program include:

- comparing numbers of people, rats and deaths in different areas of London.
- testing the equation d = (0.8 x r) x (1.3 x p) by comparing the results to historical data.
- investigating the impact of different equation weightings.

Development process

1. **Reading in data**: I began the development process by importing ‘csv’ and using it to import data from ‘death.parishes.txt’ and ‘death.rats.txt’ into ‘population’ and ‘ratscaught’ lists. In order to maintain the list structure of the files and the individual data items, I appended individual ‘people’ or ‘rat’s to a row list, then appended these lists to the

overall lists. Having the data in row lists enabled the creation of the raster map, while having it as individual data items enabled their use in calculations.

2. **Calculations:**

    a. Having read in two data sets, my next step was to generate the third data set using the provided calculation: d = (0.8 x r) x (1.3 x p). In order to maintain the structures of the raster maps, I created a for loop that looked for corresponding rows in the two data sets simultaneously using the zip function. I then employed another for loop and zip function to find corresponding pairs of individual data items within the rows, ‘p’ and ‘r’, which could be fed into the calculation. Using d = (0.8 x r) x (1.3 x p), I calculated numbers of deaths, and appended these to rows, then an overall ‘deaths’ list.

    b. In order to find the total number of deaths, I created new entities called ‘deathstotal’ and ‘deathsum’. For each row in ‘deaths’, I added the sum of its values to ‘deathtotal’. The sum of each ‘deathtotal’ was added to ‘deathsum’, providing the overall total.

    c. Using ‘csv’, I wrote code that creates and opens a new file, ‘death.deaths.txt’, and writes the generated deaths data to it row by row. The file is then closed.

3. **Visualisation:** I imported matplotlib in order to display my data as raster maps. I displayed the data using ‘matplotlib.pyplot.imshow()’, and added titles, colours and a colour-bar to improve readability. As the squares depicted locations rather than measurements, I removed the names and values on the axes.

4. **User input:**

    a. In order to allow users to interact with my program, I created a GUI using tkinter. By setting ‘root = tkinter.Tk()’, I was able to create a root window with the title ‘Model’ and some introductory text. In order to run different actions on command, I reconfigured them into functions. Some functions (e.g. ‘runpop’) called other

functions (e.g. ‘create_map’) and fed in certain attributes (e.g. the ‘population’ data). I added a menu bar with a cascade effect to my model, using ‘model_menu.add_command()’ to connect menu options with functions, enabling them to be run on command.

b. As well as enabling users to run functions, my program needed to allow users to change the parameters of the calculation. In order to request input from users, I imported ‘simpledialog’ from ‘tkinter’ to create two prompt windows asking for rat and human population weightings. The default figures were set to 0.8 and 1.3, as given in the original calculation. The ‘simpledialog’ dialog functions were assigned to entities ‘rat_inp’ and ‘pop_inp’, allowing the new figures to be recalled and used in functions. The function ‘calc_deaths’ was amended to feed in this entities and calculate: d = (rat_inp*r)*(pop_inp*p). The ‘simpledialog’ code lines were added to any function requiring the generation of deaths data. By repeating the same code structure for user input, I applied Wickens’ principle of consistency (Wickens et al, 2004, p.185), implementing the same visual and action for each user input and therefore simplifying the user experience to increase usability.

Challenges

1. **Making meaning:** A key challenge in creating my program was ensuring it was meaningful and accessible for users. Frequently, the defaults generated by Python, such as the default graph colours and axes, were not suitable for conveying the information produced by my data. By adding information to my model and graphs through meaningful labelling and the inclusion of introductory text upon launching the model, I supported users in understanding and using the program. For functions where a figure was not generated, I added message windows to inform the users of the results of the function. In particular, the message window for the creation of a txt file informed the user of the location of the new file (current directory) to enable them to find it. Finally, the raster maps were initially generated using Python’s default graph colours. These colours were not meaningful, as they did not immediately convey areas of low or high population

density, and included colours likely to cause difficulties for colourblind users (Wickens et al, 2004, p.73). Instead, I implemented a greyscale colour scheme in order to produce a more meaningful and inclusive graph. The use of increasingly dark shades to represent higher population density is in line with Wickens' principle of pictorial realism (Wickens et al, 2004, p.185). By implementing these informative and accessible elements, I was able to increase the usability of my program.

2. **Thinking through loops:** My code for calculating the estimated number of deaths involves multiple for loops. Correct indentation is therefore crucial for the code to operate as intended. Initially, my code was producing extremely large numbers of estimated deaths - these numbers were higher than the initial population numbers, making them unfeasible and indicating a mistake in my code. Having encountered a similar problem in my earlier Python project for the first assignment and solved it by adjusting my for loop indentation, I decided to apply this learning experience and investigate my indentation. In order to better understand how indentation was impacting the running of my code, I represented the process visually by drawing a diagram of what happened in each loop. By doing so, I was able to identify which loops were running too many times due to incorrect indentation and correct the mistake.

3. **User input:** I initially encountered difficulties when trying to enable user input to change the equation weightings for my program. When searching for information in the course materials, tkinter documentation, and online guidance/discussions, I had a narrow focus on the terminology from the assessment brief such as 'parameters' and 'scroll bar'. The results were largely not relevant to what I was trying to achieve, and would not be effective in my program. Instead, I decided to approach this problem by breaking the task into sections: rather than trying to tackle requesting the user input, inputting the information into the program, and changing the equation weighting all in one go, I instead looked for straightforward solutions to each stage one at a time. This approach was much more successful, allowing me to successfully implement one small stage before moving on to the next one. By breaking down the task in this way, I also made my code more adaptable, as the distinction between the different steps of the process is more visible in the program, allowing one element to be changed without impacting the others.

Sources

When I encountered challenges in the assignment, I used three types of sources to gain a better understanding of the problem and see the variety of ways in which others had approached it.

- **Teaching resources:** Firstly, I would look at the course materials and other teaching materials, such as the University of Michigan EDX course 'Getting Started with Python'[1], to deepen my understanding of the core concepts and tools related to the problem.

- **Community discussions:** Secondly, I would investigate other coders' responses to similar problems through community discussions using sources such as Stack Overflow[2]. Rather than trying to find exact matches for my challenge or focusing on specific code examples, I would look at a wide range of responses tackling related themes to find a variety of approaches.

- **Python documentation:** Having identified key concepts and approaches that might be applicable to my problem, I would look at Python documentation for particular functions that might be applicable in my project context. I would then apply the most relevant function, prioritising usability and streamlining. At times, this required trying out several functions and testing to see which was most effective.

When considering usability and accessibility in my program, I employed principles from: Wickens, C., Lee, J., Liu, Y., & Sallie, E. 2004. *An Introduction to Human Factors Engineering.* 2nd Ed., International ed. N.J. ; Harlow: Pearson Education.

Learning

Though the assessment experience I have developed my technical Python skills, with a particular focus on working with raster maps, performing calculations on data, and implementing user input. I have also deepened my understanding of good coding practice. Through tackling a

[1] https://www.edx.org/course/programming-for-everybody-getting-started-with-pyt
[2] https://stackoverflow.com/

multistage project, I have developed skills in planning and carrying out coding in smaller stages rather than approaching the project as one task. This has enabled me to focus on one function at a time, problem solve effectively by breaking challenges down into stages, and write clear, flexible code that can have elements change without impacting the entire program. I have further developed by problem solving strategies by formulating an information gathering process using academic sources and community discussions to understand key concepts and varied approaches, then looking to Python documentation to identify specific applicable functions. This approach has enabled me to draw from a wide range of concepts and strategies in order to find relevant, efficient functions to adapt and apply to my code.

Word count: 1786