4413 SUSAPY - Sustainability Analysis in Python - Reflections

Tan Chia Wu (s2823810)

14th November 2021

1. What is the most interesting new feature that you learned in this course and why?
(2 points, 1-3 sentences)

   The most interesting new feature that I learnt would be the use of git software for version tracking and recovery of unwanted changes. The software allowed me to keep regular "savepoints" of my code that I could use to revert my code in case I made an unwanted change to my code, which happened a few times during my work on the assignment. I personally used GitHub for the version control software, and it was very useful for me to learn and familiarise myself with a widely used version control software amongst coders. Using the git software also got me into the habit of making commits to GitHub after every work session.

2. What helped you most to keep an overview of your code and your progress and why (e.g. flowcharts, comments, code cells, code tags, git, etc.)?
(2 points, 1-3 sentences)

   For my code, I mainly made use of git, comments and code cells for keeping an overview of my code and progress. Besides git, which I have already elaborated in the earlier question, code cells was a very intuitive way for me to split my code up into sections corresponding to the different assignment parts, and also allowed me to work/debug on individual parts of my python script without having to run my entire script (especially importing of data and solving the logistic model for all countries). I also used comments in the code to keep track of the operations happening in small sections of code that I write.

3. What principles of fair data did you follow in this assignment and how was it implemented?
(2 points, 1-3 sentences)

   Of the 4 principles, the two which I implemented the most successfully were "Interoperable" and "Findable". As much as possible, I tried to retain as much of the Sutainable Development Goals (SDG) metadata (Indicator numbers, SeriesCode, GeoAreaCode) espcially during the export of my data into the csv file, such that my data is described with rich metadata (F2). I have also used standardised names and codes (e.g. ISO3 country codes) to index data, such that my datasets use vocabularies that follow FAIR principles (I2).

4. What was the biggest challenge you faced and how did you solve it?
(5 points, max. 150 words)

   The largest issue for my code is the mixed quality and quantity of data that was in the dataset for my SDG indicator (*SDG 13.1.1: Number of people affected by disaster (number)*). Some countries had a small number of data points (65 out of 143 countries with less than ten datapoints), while others had scattered datapoints with no clear trend. This resulted in a dataset which was difficult for me to fit the logistic or linear model to, which resulted in the code either throwing out errors or warnings. As a result, I had to resort to try/except statements to allow some blocks of code to run even when it will raise exceptions. Since running of the logistic and linear regression was in a for-loop, it was also difficult for me to track which points of data result in problems. I used print statements in the for-loop and as part of the except statements to keep track of what errors were occurring, and which country's data was responsible for the errors.

5. How did you manage to make your code more efficient and by how much? Show the code differences as displayed with git (e.g. a screenshot).
If you did not manage to make the code more efficient, reflect on which parts of the code are slow and might benefit from optimization. Moreover, show any other code differences you would like to highlight, e.g. improving the clarity, making the plot nicer, etc.
(5 points, max. 150 words + figure)

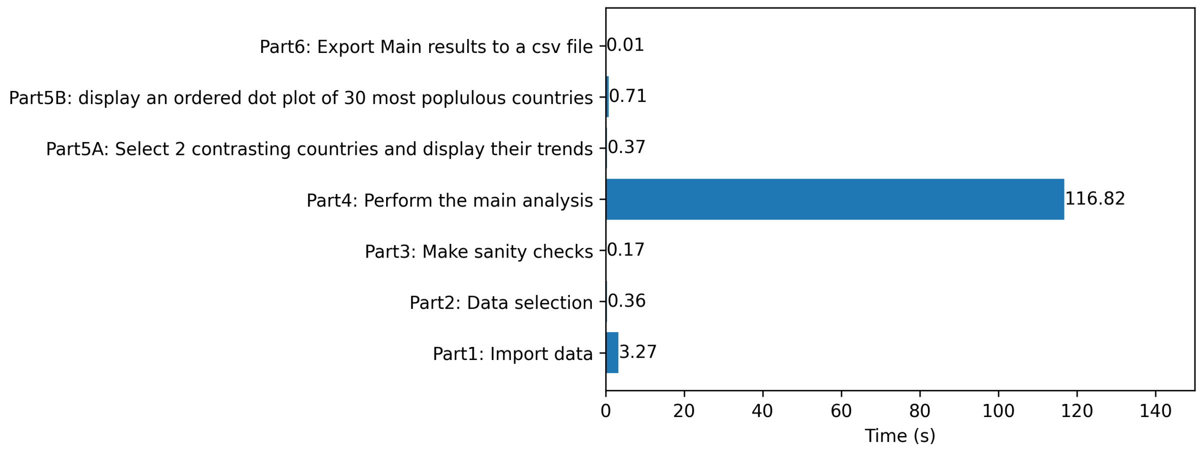
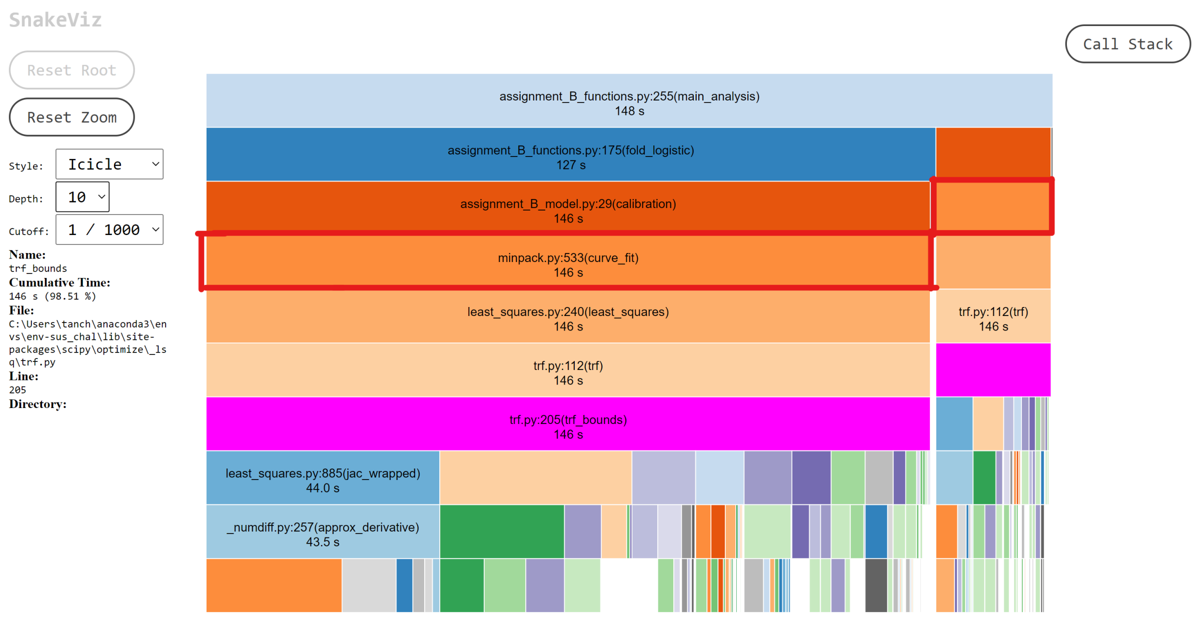


Figure 1: Comparison of the running time of each section of my `assignment_B_main.py` script.

By timing each section of code, the running of the main analysis is found to be the code block that takes the longest time (figure 1). Further analysis of the run-time of the different processes reveals that the primary contributor to the run-time of the code is the running of the trf_bounds function within the calibration function in `assignment_B_model.py` (highlighted pink in figure 2).



SnakeViz
Reset Root
Reset Zoom
Call Stack
assignment_B_functions.py:255(main_analysis) 148 s
assignment_B_functions.py:175(fold_logistic) 127 s
assignment_B_model.py:29(calibration) 146 s
minpack.py:533(curve_fit) 146 s
least_squares.py:240(least_squares) 146 s
trf.py:112(trf) 146 s
trf.py:112(trf) 146 s
trf.py:205(trf_bounds) 146 s
least_squares.py:885(jac_wrapped) 44.0 s
_numdiff.py:257(approx_derivative) 43.5 s

Style: Icicle
Depth: 10
Cutoff: 1 / 1000
Name:
trf_bounds
Cumulative Time:
146 s (98.51 %)
File:
C:\Users\tanch\anaconda3\envs\env-sus_chal\lib\site-packages\scipy\optimize_lsq\trf.py
Line:
205
Directory:

Figure 2: Results of the CProfile of the `main_analysis` function. The curve fit function (marked in red) shows up twice, first as part of the modelling step and second as part of the K-Fold validation step.

As described above in the previous question, my datapoints are scattered and this causes problems (errors and warnings) when performing regression. The provided logistic calibration function calls a curve_fit function which runs the Levenberg–Marquardt optimisation algorithm which iteratively tries to fit the logistic function to the data using a least-squares method (orange sections in figure 2). The number of iterations of the algorithm is given by max function evaluation (`maxfev`) argument

of the curve fit function. For some countries in the dataset, a solution cannot be found using the current setting of 10000.
While increasing the evaluations can resolve this error, this prolongs the run-time and runs counter to the optimisation of the code. A possible method would be filter out countries that fail the initial logistic modelling step, before running the calibration (this was not implemented), based on the assumption that the data for those countries do not fit the logistic model and can therefore be excluded from validation.

Other areas where I improved my code are integrating most of the loops into self-made functions, and reducing the use of global variables. Large sections of my code were converted into self-made functions (e.g. plotting function for observed vs simulated data) which I would call to generate plots for the top 30 most populous countries which allowed me to visualise the trends in the data easily.

6. What did you learn from reviewing the code of your peer? Did you make any changes in your code that were inspired by the code of your peer?
(4 points, max. 150 words)

The review of code of my peers was an extremely useful exercise as I took the opportunity to also challenge myself to address code problems that my peers had, or answer some of the questions in the comments of their code. Throughout this process, I applied the `pd.merge` method of DataFrames which was immensely applicable to my own code, as it allowed me to select and combine data from different DataFrames together. This was the main method I used to obtain the necessary dataset used for the .csv file and for the plotting of the ordered dot plot of the 30 most populous countries. The logic of pd.merge was also further applicable for database applications as I finally understood the different SQL joins.

Moreover, my peers approached their data from a different perspective from me, which I felt increased my exposure to the different pandas and dataframe functions and methods that are possible (groupby and sum, set theory just some that were used). Unfortunately, due to the differences in assignment and the progress of my reviewees at the point of review, I did not apply specific functions from their code.