CS5002: Programming Principles and Practice

Practical P3: Data analysis and visualisation in Python

Due date: Friday 29th November 2024 (week 11), 21:00

45% of the coursework component

MMS is the definitive source for deadline and credit details

1 Overview

The objective of this assignment is to get an experience of using Python for data analysis. By completing it, you should further enhance your Python programming skills and get more detailed insights into components of the Python ecosystem such as tools for data analytics and visualisation, and Jupyter notebooks, as well as into the practices of reproducible research using Jupyter notebooks.

This is an individual assignment.

2 Introduction

In this project you will be analysing a dataset containing 2011 census records for a random anonymous sample of 1% of Scotland's population, provided for educational purposes by the National Records of Scotland and licensed under the Open Government Licence [1]. This dataset includes records of 63,388 individuals. Each individual corresponds to a line in the CSV file containing information on 17 separate variables.

You will find the dataset in the `Scotland_teaching_file_1pct.zip` archive, which is a copy of the original dataset from [1]. It includes the following files:

- `Scotland_teaching_file_1PCT.csv` - the dataset itself;
- `Microdata_Teach_File_Overview.docx` - user guide for the dataset;
- `Teaching_File_Variable_List.xlsx` - documentation for the variables within the dataset.

3 Basic requirements

In this assignment, you are asked to do the following:

1. check the consistency of the initial (raw) data;

2. in case of any inconsistencies, output refined data for further analysis;
3. provide an executable Python script to automate the two steps above;
4. carry out certain data analysis and visualisation tasks, as outlined in the basic and additional requirements below;
5. provide a reproducible Jupyter notebook combining your report and data analysis;
6. organise code with minimal duplication for its reuse in Jupyter notebook(s) and Python scripts;
7. use version control to maintain a history of the project in a Git repository.

Your project should demonstrate the following characteristics, described in [2].

1. **Correctness**: your calculations and visualisations should be free from errors and describe the data in an accurate way;
2. **Repeatability**: you should be able to easily rerun the whole procedure from checking the consistency of the raw data to getting the final outcomes on the computer you use to work on the project;
3. **Replicability**: any other researcher (for example, the lecturer evaluating your submission) should be able to follow your instructions and replicate it on their machine;
4. **Reproducibility**: it should be easily possible to apply it to another data set with a similar structure (for example, results of the next census) and analyse the same set of metrics;
5. **Reusability**: it should be possible to reuse parts of your code in other settings.

While satisfying the first three requirements (correctness, repeatability and replicability) will be directly assessed in the process of evaluating your submission, satisfying the last two requirements (reproducibility and reusability) will be mainly judged by the project organisation and design choices.

To start with, read the dataset documentation and understand the structure of the raw data and the meaning of all information contained in the dataset. Next, create a new directory containing a plain text file called `README.txt` or `README.md` where you will later add basic details about your project. Set up the directory structure for your project, creating top level subdirectories called `data` and `notebooks`. Then put the CSV file from the archive, i.e. `Scotland_teaching_file_1PCT.csv`, into the `data` subdirectory, and read it using the provided Jupyter notebook `data/census2011.ipynb` after placing it into the `notebooks` subdirectory (for that, you will also need one more file, `data/data_dictionary.json`, placed in the `data` subdirectory).

If the notebook runs without errors, you are ready to continue this assignment. You may proceed by revising Jupyter notebooks from relevant episodes of the module, and applying the same approach to the dataset from this assignment. You will need to check that all entries match the specification, and refine them if necessary. Working with raw data, you may find useful some of the suggestions from [3], such as:

- using assertions to check your assumptions about the data;
- printing information about broken records;
- using sets or counters to store occurrences of categorical variables (when not known in advance);

- implementing ability to restart parsing in the middle of the data set (especially for much larger datasets than this one);
- testing on a small subset of your data;
- redirecting all output of a script, including `stdout` and `stderr`, to log files;
- storing raw data alongside cleaned data;
- writing verification code to check the integrity of your cleaned data.

You should use the following libraries:

- pandas (http://pandas.pydata.org/) – Python Data Analysis Library
- Matplotlib (http://matplotlib.org/) – Python plotting library

For further information, see:

- Data Carpentry lesson on data analysis and visualisation (https://datacarpentry.org/python-ecology-lesson/) for a quick hands-on introduction to pandas;
- "Python Data Science Handbook" by Jake VanderPlas has a good chapter on pandas (https://jakevdp.github.io/PythonDataScienceHandbook/);
- several tutorials on pandas are available in its documentation (http://pandas.pydata.org/pandas-docs/stable/getting_started/tutorials.html);
- another useful reference: pandas Cheat Sheet (https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf);
- finally, [4] has a collection of tips for organising your workflow in a reproducible way.

You have to use version control and track changes in your project in a Git repository. If you need an introduction to Git, you may use the Software Carpentry lesson on version control with Git (https://swcarpentry.github.io/git-novice/). Keeping Jupyter notebooks under version control, you may discover that standard tools to inspect changes and perform merges do not work well with Jupyter notebooks. To have a smooth development workflow, you may have to decide how to split your code between `.py` files (for which we suggest to use a separate directory called `code`) and Jupyter notebooks in an optimal way (this will also facilitate code reuse in executable scripts and other notebooks). You may also find useful `nbdime` – the specialised tool for diffing and merging Jupyter Notebooks (http://nbdime.readthedocs.io/).

You may use other specialised libraries as well, provided all requirements being documented in the README file, clear installation instructions are provided, and steps are undertaken to ensure the maximal portability of your code. Please ask the lecturer or the tutor in case of any concerns regarding the use of specialised libraries.

The minimal requirement for this assignment is to:

- refine the dataset:
  - develop a procedure to check that the data match expected format, remove duplicates, and perform further refinement. This procedure should ensure that:
    1. the values of variables are of the expected format (numbers, strings, etc.);
    2. the values of variables are admissible (e.g. are within a given range or are from the list of admissible values),

- in case of any inconsistencies and/or duplicates found, produce new file with refined data to be used in the subsequent analysis;
- this step must be automated to the point when it can be run with a single shell command to call an executable Python script specifying necessary argument(s);
- the refinement process should be documented (e.g. using comments in the code) in case one may need to modify and re-run it (although it's not necessary to repeat it each time while re-running the analysis),
- perform the descriptive analysis of the dataset:
  - determine the total number of records in the dataset;
  - determine the type of each variable in the dataset;
  - for each variable except "Record_Number" and "Region", find all different values that it takes, and the number of occurrences for each value,
- build the following plots:
  - bar chart for the number of records for each age group;
  - bar chart for the number of records for each occupation,
- provide the Jupyter notebook to re-run the analysis, starting from refined data;
- use version control to maintain a history of your project in a Git repository.

Important! For both basic and additional requirements, you need to ensure that the output provides textual interpretations for values of the variables from `Teaching_File_Variable_List.xlsx` instead of their alphanumeric codes. For that purpose, you have to extend and use the `data/data_dictionary.json` file.

Completing these requirements and demonstrating reasonable coding standards, efficient use of pandas and Matplotlib, accurate and informative graphics, and use of version control would be marked with the highest grade 13. In order to achieve a grade higher than 13, you should implement some of the additional requirements specified in the next section.

4 Additional requirements

Note: It is strongly recommended to ensure that you have completed the Basic Requirements and have something to submit before you attempt to deal with the Additional Requirements.

In order to achieve a grade higher than 13, you should implement some of the additional requirements below. In particular, for a grade higher than 17, you should use implement all of the requirements marked as Easy and Medium, and then attempt some Hard requirements.

Easy: build the following plots:

- pie chart for the percentage of records for each general health descriptor;
- pie chart for the percentage of records for each ethnic group.

Medium: using groupby objects, produce the following tables:

- number of records by hours worked per week and industry;
- number of records by occupation and approximate social grade.

Medium: Use pandas to find:

- the number of economically active people (see "Economic activity") depending on age;
- the number of economically inactive people (see "Economic activity") depending on a health descriptor;
- the number of working hours per week for students (codes 4 and 6 in "Economic activity").

You may use tables or plots as you would feel appropriate to illustrate your findings.

Hard: Use `ipywidgets` (https://ipywidgets.readthedocs.io/) to be able to interactively control plot properties (for example, to select a health descriptor using a dropdown list, and/or age using a slider). For a proper implementation, it is important that the plot is regenerated automatically after each use of the control elements, without the need to re-run notebook cell(s) manually.

Hard: Use `nbconvert` (https://nbconvert.readthedocs.io/) to develop a script which will produce from a command line a high quality version of your report in HTML format. Include the script and the HTML version of the report in your submission.

5 Submission

For this practical, you will have to submit a reproducible report in the form of a Jupyter notebook. This section explains how to organise this and other parts of the submission.

Submit via MMS, by the deadline of 9 pm on Friday of Week 11, a single `.zip` archive containing a plain text file called `README.txt` or `README.md`, a plain text file called `gitlog.txt`, and three top level subdirectories called `data`, `code`, and `notebooks`.

- The `README` file should briefly describe the content of your submission and provide installation instructions for the dependencies needed to run your code.
- The `gitlog.txt` should be created using Git, and should contain the commit logs of your project from its start to completion.
- The `data` directory should contain the extended version of the `data_dictionary.json` file, the raw data, and, if needed, the refined data.
- The `code` directory should contain `.py` files.
- The `notebooks` directory should contain a Jupyter notebook presenting your analysis in a form or a reproducible report, demonstrating all the outputs produced on your computer. Python source code may be distributed between this Jupyter notebook and files in the code directory imported in the notebook. In both cases, the code should be well commented.

To create the `gitlog.txt` file, navigate (e.g. in the UNIX shell on Linux and macOS, or in the Git Bash shell on Windows) to the directory of your project, and call

```
git log --oneline --graph > gitlog.txt
```

This will redirect all Git output to a `gitlog.txt` file and will properly handle outputs which need more than one screen.

You do not have to write a separate report for this assignment, since the Jupyter notebook will combine the code and its output with the project report. Instead, you should be using markdown cells in Jupyter notebook, interleaving them with code cells to describe each steps of your analysis, and also include the following:

- an introduction with the summary of the project indicating the level of completeness with respect to the Basic Requirements, and any Additional Requirements;
- a description of any known problems with your project, e.g. any Basic or Additional Requirements that are not met, but were attempted to be implemented;
- details about any specific problems you encountered which you were able to solve, and how you solved them;
- a note on how your design decisions contribute to the reproducibility and reusability of your analysis;
- an accurate summary of provenance, i.e. stating which files or code fragments were:
  1. written by you;
  2. modified by you from the course material;
  3. sourced from elsewhere and who wrote them.

If necessary, please refer to the guidelines for good academic practice as outlined in the Student Handbook.

6 Marking guidelines

Grades will be awarded according to the General Mark Descriptors in the Feedback section of the School's Student Handbook.

To give an idea of how the guidelines will be applied to this project:

- A simple prototype implementation which opens the unrefined CSV file in Jupyter notebook, reports the number of records in it and builds at least one plot, combined with a report, should get you a grade 7, and will constitute a minimum required to pass the assignment.
- A solid basic implementation which addresses all the Basic Requirements with a well commented, documented and fully working code, an evidence of use of version control, and a clear and informative report, should get you a grade 13.
- To achieve a grade between 13 and 17, you have to attempt certain number of requirements marked as Easy requirements and Medium.
- To achieve grade 17 you have to demonstrate good understanding of the basic use of version control, and implement all Easy and Medium requirements. For a higher grade you also have implement to implement some of the Hard ones.
- Providing sensibly organised, commented and documented code, together with an informative revision history in the Git repository, accompanied by a clear, informative, and appropriately structured report, with interleaved code/text cells, accurate and readable graphics, demonstrating an insight into the real world processes reflected in the data, will be crucial for getting the grades on the top of the scale.

7 Policies and Guidelines

7.1 Marking

See also the standard mark descriptors in the School Student Handbook.

7.2 Lateness

The standard penalty for late submission applies according to Scheme A: 1 mark per 24-hour period, or part thereof.

7.3 Good Academic Practice

The University policy on Good Academic Practice applies.

8 Questions?

Please let us know if you have any questions or problems. You may contact the lecturers by email cs5002.staff@st-andrews.ac.uk.

9 References

[1] Microdata teaching file and user guide. Scotland's Census. National Records of Scotland. Release date: 11 March 2021. https://www.scotlandscensus.gov.uk/documents/microdata-teaching-file-and-user-guide/.

[2] N. Chue Hong (2014): Better Software, Better Research: Why reproducibility is important for your research. figshare. https://dx.doi.org/10.6084/m9.figshare.1126304.v1.

[3] P. Guo (2016): Parse that data! Practical tips for preparing your raw data for analysis, in Perspectives on Data Science for Software Engineering, 169–173, https://doi.org/10.1016/B978-0-12-804206-9.00032-5.

[4] A. Rule, A. Birmingham, C. Zuniga, I. Altintas, S.-C. Huang, R. Knight, et al. (2019) Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. PLoS Comput Biol 15(7): e1007007. https://doi.org/10.1371/journal.pcbi.1007007.