Python Group Project
Data analysis with pandas

CS2006, 2021/22

Due date: Friday 8th April (week 11), 21:00
50% of Overall Mark for the Module

Overview

The objective of this project is to get an experience of using Python for data analysis. By completing this project, 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.

The Dataset

The dataset [4] contains information on more than 70,500 tweets with the hashtag `#CometLanding` used for the landing of the Rosetta's Philae lander on the comet 67P/Churyumov-Gerasimenko on 12th November 2014. It was collected by Ernesto Priego (Centre for Information Science, City University London, `https://orcid.org/0000-0003-4418-369X`) using TAGS 6.0 by Martin Hawksey [3].

The dataset is provided as an `.xlsx` file and may be downloaded from [4]. A copy of this file is stored in `Practicals/Python/data/CometLanding.xlsx`. In addition, the tab with the actual dataset has been exported into a `.csv` file in `Practicals/Python/data/CometLanding.csv`. It is recommended to use this `.csv` file as input data, and use the `.xlsx` only for reference.

Basic Requirements

In this project, you have to:

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. Implement unit tests to check at least some of the auxiliary functions.
5. Carry out certain data analysis and visualisation tasks.
6. Provide an executable Python script to regenerate all images and save them in a directory.
7. Provide a reproducible Jupyter notebook combining your report and data analysis.
8. Organise code with minimal duplication for its reuse in Jupyter notebook(s) and Python scripts.

Your project should demonstrate the following characteristics, described in [1]:

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 rerun easily 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 member of your project team, and the marker of your submission should be able to follow your instructions and replicate it on their machine;
4. **reproducibility:** it should be possible to apply it to another data set with a similar structure (for example, another collection of Twitter data) 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 course of marking your project, satisfying the latter two requirements (reproducibility and reusability) will be mainly judged by the project organisation and design choices (however, you may consider analysing additional datasets to demonstrate these requirements in practice).

You should first understand the structure of the raw data and the meaning of all information contained in the dataset. You do not have to make actual use of TAGS (`https://tags.hawksey.info/`) and the Twitter API (`https://dev.twitter.com/overview/api`) but their documentation is here in case you may need to consult with it. You may start from opening the CSV file using the Jupyter notebook provided in `Practicals/Python/notebooks/CometLanding.ipynb`, revising the Jupyter notebook from the lecture on exploring datasets with pandas and then applying the same techniques to the dataset from this Practical. As said in [4], "There are several duplicates in this dataset. Requires refining", so you should refine it first. At this stage, you may find some useful ideas in [2].

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`);
- Matplotlib tutorial at `http://www.labri.fr/perso/nrougier/teaching/matplotlib/` by Nicolas Rougier;
- finally, [5] has a collection of tips for organising your workflow in a reproducible way.

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 collaboration workflow, you will have to decide how to split your code between `.py` files and Jupyter notebooks in an optimal way (this will also facilitate code reuse in executable scripts and unit testing). You may also find useful nbdime – the specialised tool for diffing and merging Jupyter Notebooks, supporting Git and Mercurial (`http://nbdime.readthedocs.io/en/latest/`).

You may use other specialised libraries as well, provided all requirements being documented, clear installation instructions are provided, and steps are undertaken to ensure maximal portability of your code. You can use venv or virtualenv to have your own setup. For further instructions, see School's Wiki at `https://systems.wiki.cs.st-andrews.ac.uk/index.php/Python_on_Linux`. Please ask the lecturer or the tutor in case of any concerns regarding the use of specialised libraries.

The minimal requirement for the project is to:

- refine the dataset:
  - develop a procedure to check that the data match expected format and remove duplicates
  - 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 arguments
  - develop a set of unit tests to cover at least some of the auxiliary functions you wrote, in order to increase the robustness of your code
  - the refinement process should be documented 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:
  - calculate the total number of tweets, retweets and replies in the dataset
  - calculate the number of different users tweeting in this dataset
  - calculate the average number of tweets, retweets and replies sent by a user
  - identify most popular hashtags
- build plots/visualisations for:
  - the structure of the dataset (tweets/retweets/replies)
  - the timeline of the tweets activity
  - the word cloud for all other hashtags used in the tweets from the dataset
- provide the Jupyter notebook to re-run the analysis, starting from refined data
- provide an executable Python script to regenerate all images and save them in the `images` subdirectory

Completing these requirements and demonstrating reasonable coding standards, efficient use of pandas and Matplotlib, accurate and informative graphics would be marked with the highest grade 14. In order to achieve a grade higher than 14, you should implement some of the additional requirements specified in the next section.

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 14, you should implement some of the additional requirements below. You should not be limited by these and should feel free to discuss and implement any other feature that you consider useful (please make sure that it will be described in the report!)

- **Easy:** Analyse applications used to send tweets.
- **Easy:** Extend the descriptive analysis, for example, by calculating the average number of times each user being retweeted and the average number of times each user being replied.

- **Medium:** Analyse patterns of user activity over the period covered by the dataset.
- **Medium to Hard:** Analyse interactions between users by constructing, visualising (for example, using the networkx library, see `https://networkx.github.io/`) and analysing the graph with vertices corresponding to users and edges corresponding to their connections by means of retweets, replies and mentions. Determine some interesting properties of this graph and produce some visualisations.
- **Hard:** Implement interactive visualisiations (e.g. using sliders to control the parameters of the graph of user interactions).
- **Hard:** Analyse some other, possibly much larger, data sets.
- **Hard:** Use virtualisation tools (e.g. podman) to provide a complete environment for reproducing your experiment.

Deliverables

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 9pm on Friday of Week 11, a single `.zip` or `.tar.gz` file containing two plain text files called `README.txt` and `CONTRIBUTION.txt` file (`.md` files are also acceptable), and three top level subdirectories called `data`, `code` and `notebooks`.

The file `CONTRIBUTION.txt` should be prepared individually by each member of the group. It should contain your matriculation number and describe your own contributions to the project.

The rest of the submission should be the same for everyone in the group.

The `README.txt` file should briefly describe the content of your submission and provide installation instructions for the dependencies needed to run your code.

Furthermore, the `data` directory should contain the data (raw and, if needed, refined), and the `code` directory should contain the code in the form of `.py` files. There may be further directories and subdirectories, and other files, if necessary.

The `notebooks` directory should contain a Jupyter notebook with your analysis. 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.

Furthermore, the `data` directory should contain the data (raw and, if needed, refined), and the `code` directory should contain the code in the form of Jupyter notebooks and supplementary `.py` files if needed. The `images` directory is a place to hold a collection of images generated by a Python script. There may be further subdirectories and other files, if necessary. The Python source code may be contained in Jupyter notebook or in separate files that the notebook will import, but in both cases it should be well-commented.

There is a special procedure for the submission of Jupyter notebooks: for each notebook, you should provide both an `.ipynb` file with all outputs, located in the `code` directory, and its export to PDF format, located in the `report` directory. The marker should be able to use the `.pdf` file as a reference, clear all outputs in the `.ipynb` file and rerun it to produce the same output as shown in the `.pdf` file.

The recommended procedure to export a Jupyter notebook to PDF is to print it to PDF using the browser Print menu. It was observed that Chrome may produce better output than Firefox. It is recommended to check the output to ensure that all critical details remain readable in the submitted PDF file. **DO NOT** use "File" → "Download as" → "PDF via LaTeX" option in Jupyter.

You do not have to write a separate report for this Practical, 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 the level of 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.

Also, remember that the place to document your individual contributions is the `CONTRIBUTION.txt` file in the root directory of your submission.

Marking Guidelines

This practical will be marked according to the guidelines at `https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html`.

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

- A simple prototype implementation which opens the CSV file in Jupyter notebook and reports the number of records in it, combined with a report, should get you a grade 7.
- A solid basic implementation which addresses all the Basic Requirements with a well commented, documented and fully working code, combined with a clear and informative report, should get you a grade 13.
- To achieve a grade between 13 and 17, you have to implement between one and two requirements marked as **Easy** and between one and two requirements marked as **Medium**.
- To achieve a grade higher than 17 you have to go beyond the requirements described above by either suggesting and implementing additional Easy and Medium requirements, or by implementing some of the Hard ones.
- Providing sensibly organised, commented and documented code, 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.

In addition, remember that:

- Standard lateness penalties apply as outlined in the student handbook[1].
- Guidelines for good academic practice are outlined in the student handbook[2].

Finally

The project is very much open-ended, so please feel free to discuss your own agenda in addition to implementing Easy and Medium requirements in order to score a high grade. Be creative, have fun, and produce something which you would be interested to make and would be proud of!

Please let me or your tutor know if you have any questions or problems. You may contact me by email `obk1@st-andrews.ac.uk` or via Microsoft Teams.

Olexandr Konovalov
March 2022

[1] `https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html`
[2] `https://info.cs.st-andrews.ac.uk/student-handbook/academic/gap.html`

References

[1] 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.

[2] 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

[3] M. Hawksey: TAGS 6.0, https://tags.hawksey.info/

[4] E. Priego. A #CometLanding Twitter Archive. City, University of London (2014), https://doi.org/10.6084/m9.figshare.1271659.v3.

[5] 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