COSC 2670/2732 Practical Data Science with Python

Project Assignment 1, Semester 2, 2021

Marks : This assignment is worth 30% of the overall assessment for this course.

Due Date : Fri, 20 August 2021, 11:59PM (Week 5), via `canvas`. Late penalties apply. A penalty of 10% of the total project score will be deducted per day. No submissions will be accepted 5 days beyond the due date.

Objective

The key objectives of this assignment are to learn how to process messy text data using python. Data is always going to start in a form you may not want. So, you must massage the data into a form you do want. Python is a perfect tool for processing large volumes of raw data.

Australians love sports, so it seems fitting to begin our data adventure by processing a bit of statistical data from Australian Rules Football. If you are unfamiliar with the sport, you might want to watch this video `https://https://youtu.be/Dtmu-1kMFZw` and/or read a bit about the rules on Wikipedia `https://en.wikipedia.org/wiki/Australian_Football_League`.

Provided files

The following template files are provided:

- **A1.ipynb** : a skeleton Jupyter notebook file you can use to interactively work through your initial testing to implement the functions as defined below.
- **create_afl_tsv.py** : a skeleton main file for Challenge 1. Once you have got your code working in the Jupyter notebook, you should move the code you need here so that it will "pass" a set of automated tests that we will run to verify your code can produce valid outputs for each of the function challenges you have been given.
- **validate_afl_tsv.py** : a skeleton main file for Challenge 2. Same as above.
- **find_top_wins.py** : a skeleton main file for Challenge 3. Same as above.
- **aflutil/helper.py** : Several helper utility functions that are provided to read / write files in a specific format. You should look at all of these functions carefully as they will help you solve the challenges.
- **aflutil/AFLGame.py** : This is a helper class that is used only in the third challenge.
- **data directory** : a set of markdown data files containing the input data to be used for this project.
- **requirements.txt** : this file should be used by you to ensure that you work in a clone of the anaconda environment. This should not be modified. It is critical that you use the environment as defined or you will fail the automated test we will be running to validate your project code. See below for further instructions.

The Templates

We realise that many of you are just getting used to python, and there is much to learn. So we have tried to provide you a well-defined harness to guide you towards a working prototype. Once you have the data, you will apply some basic data analysis techniques to find useful information in the data.

Creating Anaconda Environment

The first task is to create the correct Anaconda working environment. The rules for this may differ a little depending on every OS, but you should be able to find the right invocation for your platform of choice. I show the *command line* invocation here.

```
conda create -n PDSA1 python=3.8
conda activate PDSA1
pip install -r requirements.txt
```

That's it. This will create a new environment you can start in Anaconda using "`conda activate PDSA1`" and exit from using "`conda deactivate`". For the first assignment you should not need anything except for the python core library and the packages related to Jupyter notebook.

The Data

The data you will be processing is data that has been crawled from the web to produce a set of markdown files. The data is raw statistical information for Australian Rules Football. You should study several of the input files in a text editor to get a feel for what you will have to parse. You will quickly start to recognize clear patterns in the line structure that you will exploit to quickly process thousands of lines of raw data. The two main file types are team-based statistics. The data is not clean, but it is well-formed. So this makes it very amenable to python data wrangling to extract and aggregate all of the data into a more usable form – a dataframe. `pandas` is a popular Python package that is used regularly in data science. So you will find that dataframes are a very useful way to organize and process columns of data similar to a database table – without all the overhead of a full rbdms system. However, as many of you learning python still, we will not use a dataframe in this project. It would be relatively easy to convert the data array we are using in this project into `pandas` as you will have done all the hard work of cleaning the data, but we will save that challenge for another day.

Program output

When your program is combined with the datasets supplied, your program should produce the output as specified in the code template. The primary output format is actually a `tsv` file which is similar to a `csv` file, but uses tabs instead of commas to separate fields. When working with long strings of text, you are more likely to avoid collisions as tabs can easily be removed from a text file by replacing them with spaces, but not commas. Once you have `tsv` files (or even `csv` files), it is easy to serialize them out to a file for storage and reload them when you need the data again. You can also easily load some and not all of the data if there is a lot to process too. Do not change the output functions provided, or you will fail to pass the automated harness tests. All you need to do is to implement the functions in the skeleton code. Once you get each of these functions to work, it will output the answers automatically. Each function is worth a subset of the the 30 possible points you can get on the project.

Processing Raw Team Statistics (15/30 marks)

The first challenge is the core of our first assignment. The basic idea is to process several semi-structured text files which contain outcomes for all AFL games for each team. For example, the start of the `richmond.md` file contains the following information:

```
| --- |
| Richmond |
| --- |
|  2021 |
| --- |
| Rnd | T | Opponent | Scoring | F | Scoring | A | R | M | W-D-L
      | Venue | Crowd | Date |
| R1 | H | Carlton | 3.3 8.5 10.8 15.15  | 105 | 3.2 6.6 8.12 11.14
      | 80 | W | 25 | 1-0-0 | M.C.G. |  49218 | Thu 18-Mar-2021 7:25 PM |
...
```

There are only two fielded line types, the ones containing header information like the first four, or the core statistics lines with 14 fields of data. Our goal is simply to walk each of these files and create a single two-dimensional array (list of lists in python). Each line of the array will contain the following 15 cells:

1. Team - Will be the the name and the second line of every file.
2. Year - A header field that will come before all of the round statistic lines.
3. Round - Round or Playoff Round
4. Where: T above and is an H (home), A (away), or F (final/playoff)
5. Opponent
6. For_Scoring - quarterly scoring like "1.3 5.4 8.6 13.12". Here, 1 represents 1 goal scored and 3 behinds in the first quarter. These are cumulative, and will be used to verify scores.
7. For_Total - total score by this team (F above).
8. Against_Scoring - this is the same as For_scoring but for the opponent.
9. Against_Total - opponent total (A above)
10. Result - Final outcome, which is a W (win), L (loss), or D (draw).
11. Margin - difference between the two total scores.
12. WDL - Current record of this team (not the opponent).
13. Venue - stadium for game.

14. Crowd - size of crowd.
15. Date - date and time of game.

So, we can see that the columns we really want with the exception of team name and year which you can extract easily from the files are already the way we want them to be – we just need to split the lines into columns. We will talk in the lectorials and in practicals how to process a text file line by line as well as locate and split the lines we want using the `line.split('|')` command. This may seem daunting at first, but once you get the hang of it, you will find that parsing data files is surprisingly easy in Python. You just have to know what you need to extract and set up guards to ensure you ignore the lines you do not care about. You will want to ignore the header lines which repeat for each year and ignore two lines of cumulative statistics at the end of each year. These lines will look like this:

```
| Totals | 188.162 |  1290 | 187.177 |  1299 |
  P:16 W:7 D:0 L:9 |  | 577583 |  |
| Averages |  12.10 |  81 |  12.11 |  81 |  |  |  36099 |  |
```

So if the first column contains `Rnd`, `Totals`, or `Averages`, you want to skip over it. You will skip the separator lines like `| --- |` too. Everything else you will want to carefully capture as you walk each file in order to build complete rows in the final array.

Challenge 2 Validate total scores, margins, and outcome (10/30 marks)

This task will challenge you a little more. Several rows of data are not guaranteed to be correct in the raw data. However, there is always enough information in a row to validate and correct the errors you encounter. More specifically, the total scores for home and away teams may be incorrect, which means the margin and final outcome may be wrong too. However, the quarterly scores for both home and away teams are always correct, so your goal is to parse these two fields and separate off the last recorded quarter score. Since these are cumulative, it is all you need to get the final score for the team. For example, given a game scoring of “1.0 1.4 4.5 5.8”, you would separate off “5.8”, split 5 and 8, ensure they are integers and not strings, multiply 5 by 6 as each goal is worth a total of six points, and add 8 as each behind is worth only one point.

In summary, you need to check every `For_Total`, `Against_Total`, `Margin`, and `Result` column in every row to ensure that the scores are all correct, the margins are correct, and the final outcome is correct. All changes used to update the current array and it is written out one last time.

Challenge 3 Find top five wins home and away (5/30 marks)

This increases the bar one more time and will require you to use a non-trivial data structure / function to find the biggest wins of all time – home and away. This challenge is definitely easier if you know you have the correct margins for every game. If you do, you just need to find the rows with the largest margins for home wins and away wins in the data set.

Hint: There are definitely multiple ways to achieve this goal, but we will cover an example in the lectorials that shows a problem that is analogous to this one. So pay attention in lectorials and make sure you understand the `heapq.nlargest` call when we cover it.

Submission

All submissions must be made through MyRMIT. A link will be provided within canvas, under the Assignments tab for submissions. Assignments submitted through any other method will not be marked. To submit your files, create a top level directory which is your student number. Inside that folder, there should be exactly like the zip file provided to you. You **should not submit the data directory**. Do not change these files as I will be using only the original to verify your output. So you do not need to submit them. They will just slow down your upload or even cause it to fail. So move it out of the way when you create your submission zip. You will make your life difficult if you don't!

So in other words – Rename the folder A1 to your student number and zip up that directory for submission. Move the data directory somewhere else when you are ready to submit. For example, if your student ID is S101010, when I unzip the file `S101010.zip`, I would see the following:

```
$ unzip S101010.zip
Archive:  S101010.zip
   creating: S101010/
  inflating: S101010/create_afl_tsv.py
  inflating: S101010/validate_afl_tsv.py
  inflating: S101010/find_topfive_wins.py
```

The following files should be submitted (Please do **not** submit any test files):

- `create_afl_tsv.py`
- `validate_afl_tsv.py`
- `find_topfive.py`

NOTE 1 : All submissions should use the anaconda environment provided. Trust me, if you do not, you will probably fail. Python is notorious for changing APIs in packages. The best way to ensure your code will run is to carefully create and freeze your development environment. You can always update over time, but as an experienced Python developer I have learned to be careful when I do. It is very unforgiving if you don't know exactly which package versions and python release you were using when you developed your code. Anaconda is an amazing tool to help you configure and control a single environment for each project. So spend time getting comfortable with `conda`. It is time well-spent if you want to use Python on important projects.

NOTE 3 : There is a discussion group available in the course canvas. Please do not post code snippets in this forum. Do ask questions if you have them! We will do our best to answer them as quickly as we can.

Plagiarism Warning

University Policy on Academic Honesty and Plagiarism: It is University policy that cheating by students in any form is not permitted, and that work submitted for assessment purposes must be the independent work of the student concerned. Please see the RMIT policy for more details: `http://rmit.info/browse;ID=sg4yfqzod48g1`. **THIS IS NOT A GROUP PROJECT.** Students are reminded that this assignment is to be attempted individually. Plagiarism of any form will result in zero marks being given for this assessment, and can result in disciplinary action. We routinely use plagiarism software on projects! Please, please don't do it. Be aware that paying someone on a coding site to do it for you is a form of plagiarism. If you are submitting work that is not your own, regardless of how you got it – you are in breach of this policy.

Extension Policy

Individual extensions will **NOT** be considered or granted by the PDS team. We are not monsters, but doing things on time matters. We consider it a core component of the assessment. If we decide to extend the deadline, the only fair way we can do this is to extend it for *everyone*, and that may not be possible given the the size of the course and university policies. We have set deadlines as late as we can in order to meet the university timelines we cannot control. So be early and not late. If you procrastinate and suddenly have other priories, you will be in a real bind, and that is not a valid reason for an extension.
If you have suffered a personal tragedy or illness, there is a University process in place to grant extensions. Our preference is that you go this route if you must, as they have very clear criteria to grant exemptions. For more information about applying for Special Consideration, see the rules and regulations at `http://www.rmit.edu.au/browse;ID=b1wqvnwk8aui`.

Getting Help

Come talk to us. Email us. Use the discussion board. Ask a question in a lectorial or a practical. There is help available if you need it.