Fantasy 5 Data Analysis

Author by Donald Cao
An independent project
Written on 03/15/2022

Summary

Running a python script to extract and clean desirable data from APIs. This method allows another approach to web scrape should HTML or bot scraping isn't possible. Also, an introduction to a python package glom for acquiring deep nested data by dot notation.

Introduction

There had been cases where data extraction had gone not expectedly well. In the case of CALottery.com, this dynamic website was a disappointment. If approaching with the usual web scraping tools like Beautiful Soup or Scrappy, they would fail extracting past winning draws. This is due to the data being stored and represented in a JSON scripted file. Therefore, hinder the attempt with these tools. But at last, the use of developer tool's Network provide the JSON files' URL Address and the use of glom had help extract the deep nested data and further progress with data clean up and statistical analysis.

Procedure

Navigate to the webpage of interest, https://www.calottery.com/draw-games/fantasy-5#section-content-2-3 and read the privacy policy, term of service, and find the robot.txt file if available. This would give clues if the website is legally able to be scraped. As of writing this, it states that if it's for educational purpose and if no harmful intention is done with the data, then it would be allowed. To stress, READ THE DOCUMENTS (Private Policy, Term of Service, Robot.txt) to abide to the law – for your own safety.

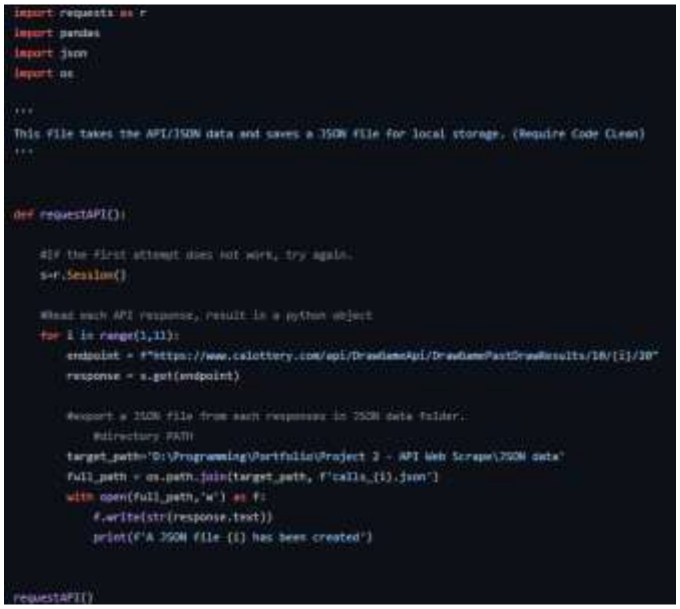
While having the webpage open, click on the tab "Past Winning Number". This would open the tab and show all previous draws. Then right-click anywhere within in the tab to open a box and select "Inspect" or "Developer Tool".

Now, go to "Network" and select Fetch/XHR tab. Then refresh the webpage. This will show the scripted file being obtain and displayed on the webpage as an object in real-time. The JSON file name is "20". There will be duplicate of it, so select whichever one and continue.

I do not endorsed gambling from this article, it is purely for entertainment and educational purpose only. Please play at your own risk, discretion is advised.

Right-click the file "20", go to "Copy", and Select "Copy link address". This would be use as your locator on python.

Having now the URL, open a new python and title it "extraction.py". Your program script will look like:

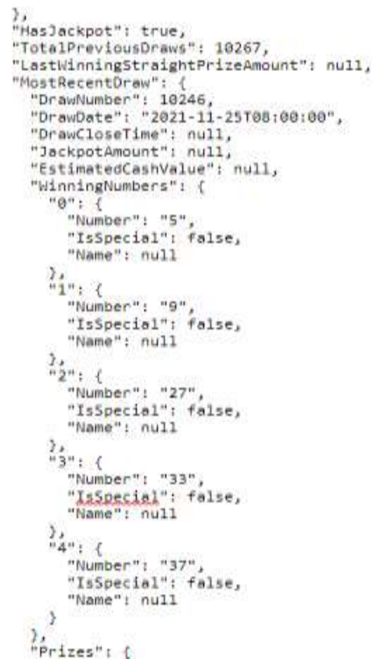


```python
import requests as r
import pandas
import json
import os

'''
This file takes the API/JSON data and saves a JSON file for local storage. (Require Code Clean)
'''


def requestAPI():

    #If the first attempt does not work, try again.
    s=r.Session()

    #Read each API response, result in a python object
    for i in range(1,11):
        endpoint = f"https://www.calottery.com/api/DrawGameApi/DrawGamePastDrawResults/10/{i}/20"
        response = s.get(endpoint)

        #export a JSON file from each responses in JSON data folder.
            #directory PATH
        target_path='D:\Programming\Portfolio\Project 2 - API Web Scrape\JSON data'
        full_path = os.path.join(target_path, f'calls_{i}.json')
        with open(full_path,'w') as f:
            f.write(str(response.text))
            print(f'A JSON file {i} has been created')


requestAPI()
```

Please note that there will be multiple files to account, thus the use of loop is used. Save as all JSON files in a dedicated folder explicitly for them.

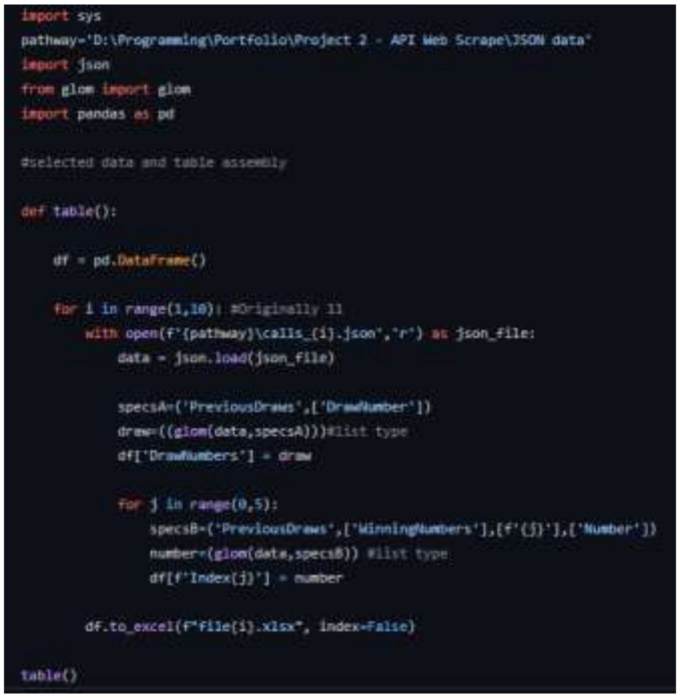
With having these JSON file saved locally, its easy and faster to access them with python. And now is time to wrangle these data, to select and clean these raw data into useable data for statistical analysis.

As you can see, the JSON has nested data:



```
},
"HasJackpot": true,
"TotalPreviousDraws": 10267,
"LastWinningStraightPrizeAmount": null,
"MostRecentDraw": {
  "DrawNumber": 10246,
  "DrawDate": "2021-11-25T08:00:00",
  "DrawCloseTime": null,
  "JackpotAmount": null,
  "EstimatedCashValue": null,
  "WinningNumbers": {
    "0": {
      "Number": "5",
      "IsSpecial": false,
      "Name": null
    },
    "1": {
      "Number": "9",
      "IsSpecial": false,
      "Name": null
    },
    "2": {
      "Number": "27",
      "IsSpecial": false,
      "Name": null
    },
    "3": {
      "Number": "33",
      "IsSpecial": false,
      "Name": null
    },
    "4": {
      "Number": "37",
      "IsSpecial": false,
      "Name": null
    }
  },
  "Prizes": {
```

Open a new python and save as dataframe.py. The purpose is to extract selected data and organize it into a data frame or similar to a table. To access the deep nested data from JSON, the use of Glom is necessary. Therefore, the code will be:



```
import sys
pathway='D:\Programming\Portfolio\Project 2 - API Web Scrape\JSON data'
import json
from glom import glom
import pandas as pd

#selected data and table assembly

def table():

    df = pd.DataFrame()

    for i in range(1,10): #Originally 11
        with open(f'{pathway}\calls_{i}.json','r') as json_file:
            data = json.load(json_file)

            specsA=('PreviousDraws',['DrawNumber'])
            draw=((glom(data,specsA)))#list type
            df['DrawNumbers'] = draw

            for j in range(0,5):
                specsB=('PreviousDraws',['WinningNumbers'],[f'{j}'],['Number'])
                number=(glom(data,specsB)) #list type
                df[f'Index{j}'] = number

        df.to_excel(f"file{i}.xlsx", index=False)

table()
```

These data frame tables are now saved as csv for easy access and saved in another dedicated folder.

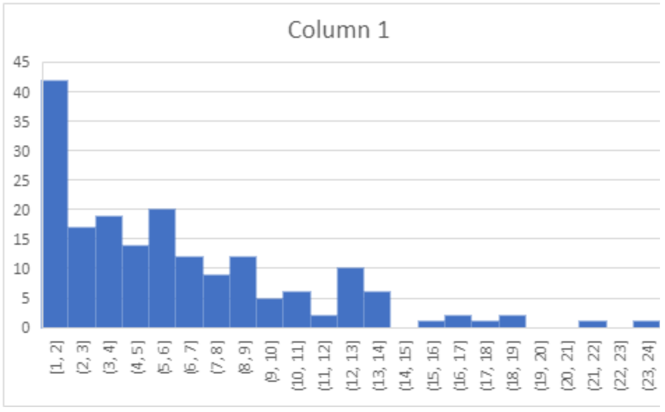
Results

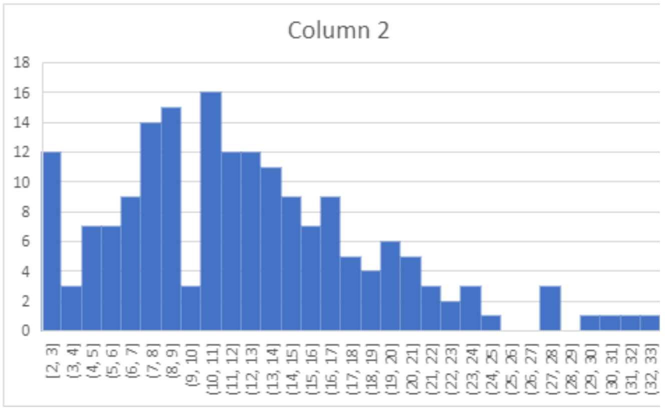
From this point on, you can go either direction:

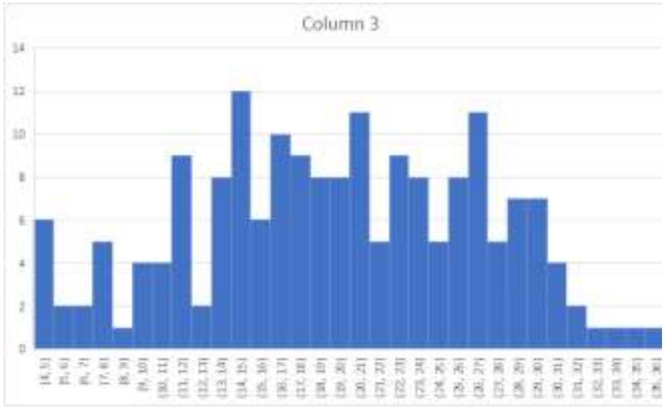
1. Approach with python purely, using Pandas, Numpy, and Matplotlib. This would gives you more control of draw an accurate visualization and account data for Descriptive Analysis.
2. Convert manually all the csv file into xlxs and then proceed with the Descriptive Statistics.

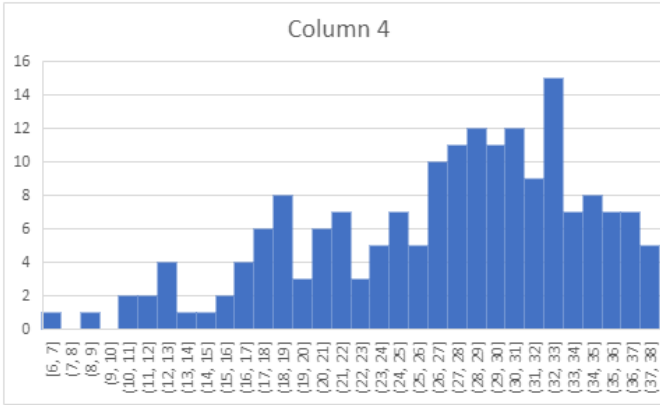
In the case for this article, lets go with the latter. Avoid the hassle of writing too much code and go for the statistics immediately! This means, manually coalescing all ten files and analyze each number columns to see the results.

Drawn	Col.1	Col.2	Col.3	Col.4	Col.5
10346	22	28	33	35	36
10345	9	18	24	30	32
10344	4	16	21	28	33
10343	1	2	4	11	19
10342	1	3	9	21	29
10341	14	17	20	23	38
10340	5	16	27	33	35
10339	3	13	19	26	27
10338	2	11	23	35	37









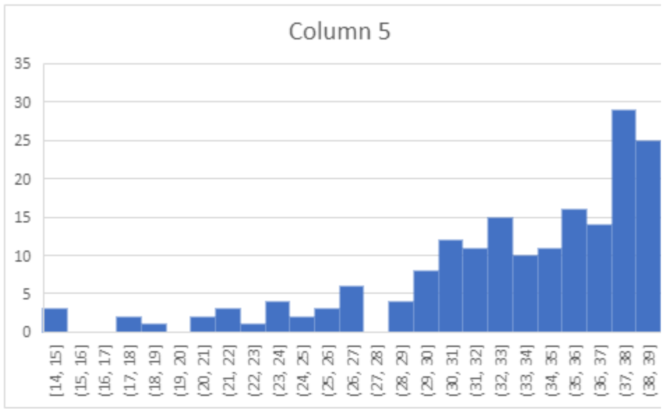


Fig 1a-e: Shows the table of the data along with the histogram of each number and their frequency by their associating position.

Conclusion

Witnessing the visual graphs of the findings, this process complete and can be said that the process of extracting the data through API is possible. And all it took was locating the API file and python.

Discussion

This was the best I can approach I can make for an attempt at data engineer with script and dynamic website.

If I could had done this differently, I would have continued on from the csv files and utilize SQL. But I also would have added more attributes, like date of drawn and add an index. This would help not only keep a persistent storage, but also enable the data set to be retained every time a new call is made. Sadly, with still very few attributes, using an RDBMS is inconvenience, thus the use of xlxs is used instead.

Despite having to end the statistic analysis only at the descriptive statistics, you can try and follow up with a project to predict the winning draws. That is if you choose to. But looking back, the total drawn set was a total of 202 calls from all ten JSON files. As new drawn set is called daily at six in the evening and posted by midnight, the oldest of the previous drawn set is removed. This means, data set is incomplete and seen as a small sample size. This greatly limits the predictive models available to test. This means the only predictive models for exploratory statistics is conditional probability follow with a Bayes Theorem for validation test. Another approach is regression line test, but that model would not fit with this data.

Overall, the process to extract and clean data was a success and this project is considered completed for the purpose of this run. Though, it does leave room to be satisfied and preferably to see if further improvement can be done. But as stated earlier, SQL would not be useful due to the lack of attributes. Also, if the test was a longitudinal test, manually updating the xlxs file with daily winnings, would be redundant and not make a difference in the frequency histogram. With that said, what would you had done differently than I had done?