Web Scraping III

ISTA 350 Hw8, Due 4/30/2020 at 11:59 pm

Introduction. This homework is the third part of an introduction to the art of web scraping. Now that you have scraped, cleaned, and stored some data, you will do some statistics on it and plot it. At the heart of this assignment are the `numpy`, `pandas`, and `matplotlib` modules.

Instructions. Copy your `hw7.py` module and rename the copy `hw8.py`. Below is the spec for five new functions and some additions to `main`. Implement them and upload your module to the D2L Assignments folder.

Your plots need to exactly match the provided exemplars. Correct versions of hw8's output are also provided in case you didn't get hw8 to work correctly. More details are in the documentation at the top of the test file.

Testing. Download `hw8_test.py` and the associated files necessary for testing and put them in the same folder as your `hw8.py` module. Run it from the command line to see your current correctness score. Each of the five new functions in `hw8.py` is worth 20% of your correctness score. However, the test file will only grade three of them. The SL's will check the results of your `make_plot` function by hand. Thus, the test file only reports how your program fares on 60% of the assignment. The test file we will use to grade your program will be different and may uncover failings in your work not evident upon testing with the provided file. Add any necessary tests to make sure your code works in all cases.

Documentation. Your modules must contain a header docstring containing your name, your section leader's name, the date, `ISTA 350 Hw8`, and a brief summary of the module. Each method/function must contain a docstring. Each docstring should include a description of the function's purpose, the name, type, and purpose of each parameter, and the type and meaning of the function's return value.

Grading. Your module will be graded on correctness, documentation, and coding style. Code should be clear and concise. You will only lose style points if your code is a real mess. Include inline comments to explain tricky lines and summarize sections of code (not necessary on this assignment).

Collaboration. Collaboration is allowed. You are responsible for your learning. Depending too much on others will hurt you on the tests. "Helping" others too much harms them in reality. Cite any sources/collaborators in your header docstring. Leaving this out is dishonest.

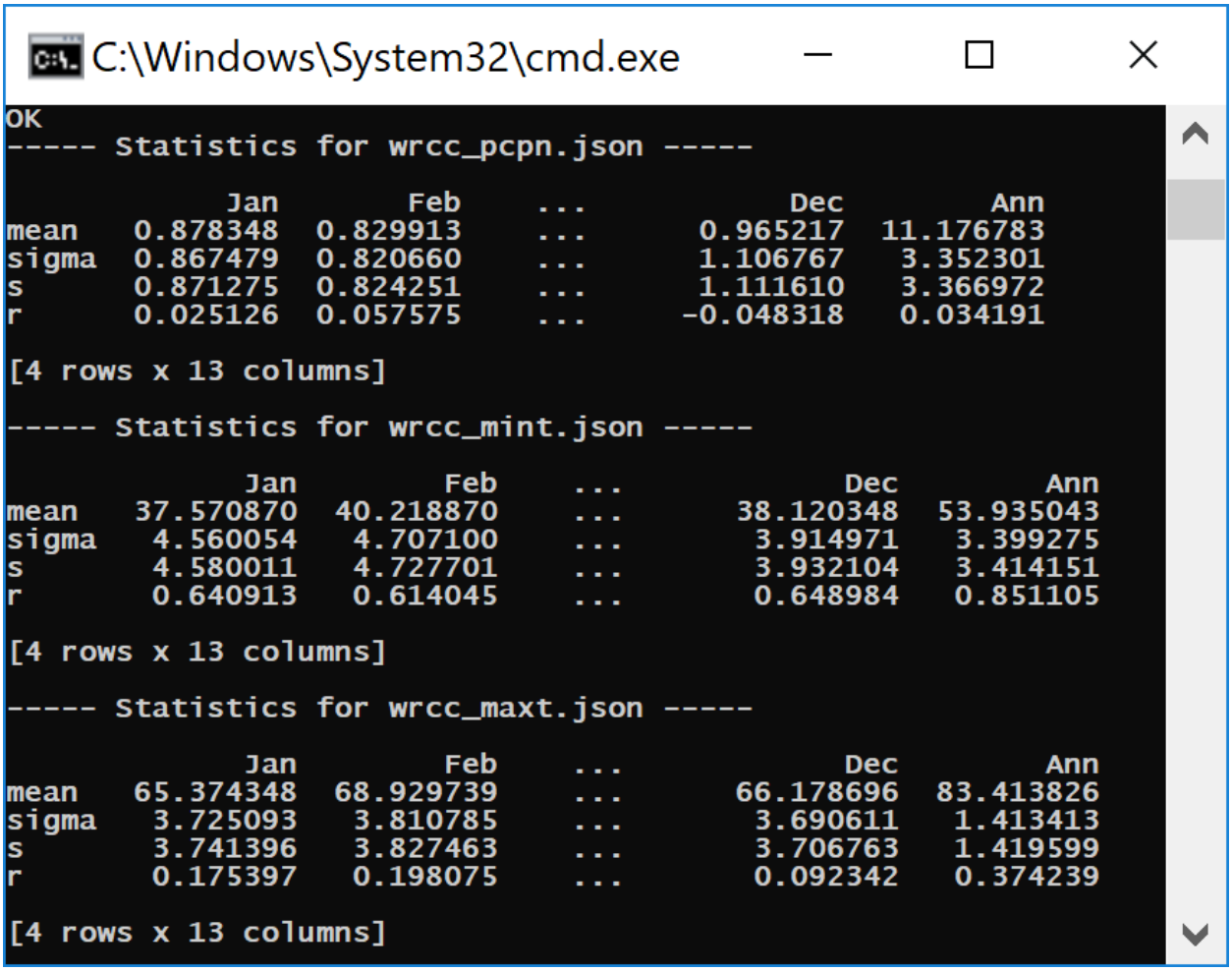
Resources.
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

`hw8.py`:

`get_panda`: this function takes a string representing a filename as its sole argument. The file contains a `json` object representing a list of lists. Load the list and store it into a variable. Return a `DataFrame` that has the same data as the list with row labels (`index`) that are the three-letter abbreviations for the months and the year `['Jan', 'Feb', …, 'Dec', 'Ann']` and column labels (`columns`) that are integers representing years `[1894, 1895, …]`.

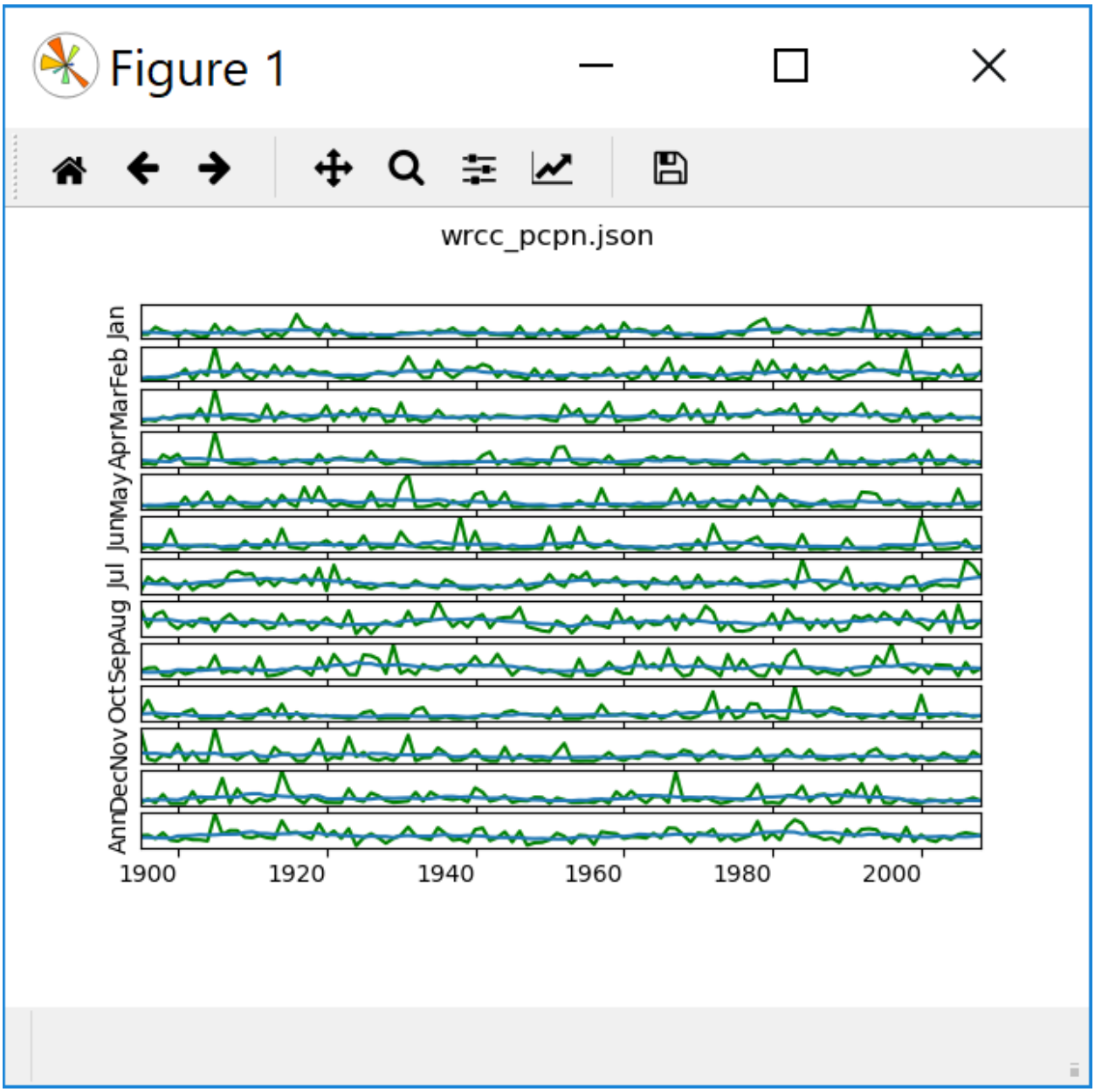
`get_stats`: this function takes a `DataFrame` as its sole argument and returns a new `DataFrame` containing a statistical summary of the argument. Use these values for the return object's `index`: `['mean', 'sigma', 's', 'r']` and the `index` from the argument for its `columns`. Calculate the statistics and populate the `DataFrame` with them. You may want to store them in a list of lists or `np` array and then pass that to the `DataFrame` constructor.

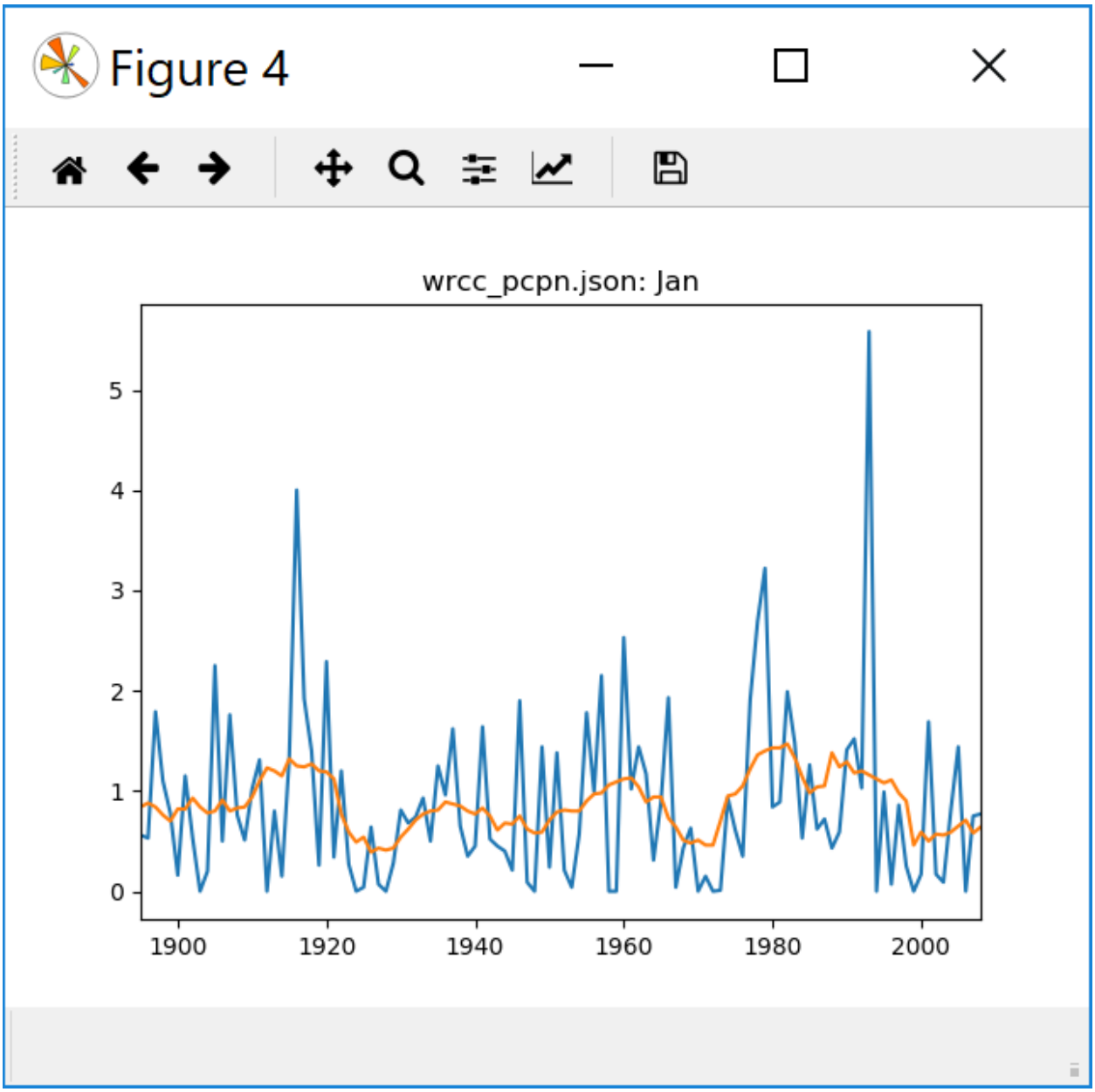
`print_stats`: this function takes a filename as its sole argument. Print the following header: `'----- Statistics for <filename> -----\n'`. Create a `DataFrame` from the data in the file. Print a `DataFrame` containing a statistical summary of that data. Print a blank line. Printing may differ by platform, but your numbers should be really close and print something like this:



`smooth_data`: this function takes a `DataFrame` as its first argument and returns a `DataFrame` with the same `index` and `columns` but each data point has been replaced with the 11-year average of the surrounding data including the data point itself. The second argument specifies a precision for each datum (number of decimal places) in the new `DataFrame` and has a default value of `5`. For example, the minimum temperature for March, 2000 will be replaced by the average of the minimum temperatures from March, 1995 to March 2005. If there are not enough years to go five years out one way or the other from the central data point, use the available years.

`make_plot`: this function takes three arguments – a filename, the three-letter abbreviation for a month, and a precision. The second parameter has a default argument of `None`; the third has default of `5`. Make a `DataFrame` from the file and use that to make a `DataFrame` of smoothed data. In order to use the pandas plotting functionality, you need to transpose both `DataFrames`. If no month argument was passed, plot all of the data in the argument, each month with its own subplot, no legend or yticks, and make the title the name of the file the data came from. Label each subplot on the left with the three-letter abbreviation for the month the data represents. Plot the smoothed data for each month on the appropriate subplot. If the month argument isn't `None`, plot just that month's data and its smoothed data, with the title being the filename followed by a colon and the month string. The plots (you can look at the saved figs for more up-close inspection) should look like this:





`main`: In addition to your other imports, you will need this line:

```
import matplotlib.pyplot as plt
```

add these lines to your `main`:

```
for fname in fnames:
    json_fname = fname.split('.')[0] + '.json'
    print_stats(json_fname)
    make_plot(json_fname, precision=2)
plt.figure()
make_plot(fnames[0].split('.')[0] + '.json', 'Jan')
plt.show()
```

Now you have a complete program that can go to the web, retrieve the data, save it in various formats, clean it, analyze it, and plot it. Once you have it working, consider these questions: **Do your statistics**

match those on the website? If not, why not? Did they do sample or population statistics? Can you tell? What does your data tell you about climate trends at UA? What factors might account for those trends, if you found any? What factors might obscure your ability to ferret out any trends? You calculated both population and sample standard deviations for the data. Which of these was more appropriate? Why?