Arctic Sea Ice Extent: Data Visualization

ISTA 131 Hw8, Due 10/21/2019 at 11:59 pm

Introduction. This homework is the second in a two assignment arc intended to introduce you to dealing with data that is stored on disk in csv format using pandas. In this assignment, you will reproduce figures from the web created by the National Snow and Ice Data Center (NSIDC) and Japan's version, the Arctic Data Archive System (ADS). The NSIDC plot is the way they used to do it, better than their current version. The ADS site will be changing soon, so who knows how long that link will work. Plus, they made their plots worse. We'll recreate the old style.

Instructions. Create a module named `hw8.py`. Below is the spec for seven functions. Implement them and upload your module to D2L Assignments.

Testing. Download `hw8_test.py` and auxiliary testing files and put them in the same folder as your `hw8.py` module. Each of the first three functions is worth 20% of your total score. You do not directly receive points for `main`, but it has to work correctly or you will get no points for two of your other functions. You can examine the test module in a text editor to understand better what your code should do. The test module is part of the spec. 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 module must contain a header docstring containing your name, your section leader's name, the date, `ISTA 131 Hw8`, and a brief summary of the module. Each 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.

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.

http://nsidc.org/arcticseaicenews/
https://ads.nipr.ac.jp/vishop/#/extent
http://pandas.pydata.org/pandas-docs/stable/api.html
https://docs.python.org/3/tutorial/datastructures.html
http://matplotlib.org/users/mathtext.html
http://matplotlib.org/api/axes_api.html

`get_column_labels`: Cut-and-paste this one from hw7, if you want to use it.

`get_2019`: This function reads the data from `data_2019.csv`, which looks like this:

	A	B
1	1/1/2019	12.876
2	1/2/2019	12.956
3	1/3/2019	12.959
4	1/4/2019	13.047
5	1/5/2019	13.098

and returns a `Series` that looks like this:

```
In [5]: y2019.head()
Out[5]: 0101    12.876
        0102    12.956
        0103    12.959
        0104    13.047
        0105    13.098
        dtype: float64

In [6]: y2019.tail()
Out[6]: 1013    5.109
        1014    5.150
        1015    5.224
        1016    5.262
        1017    5.374
        dtype: float64
```

I used `get_column_labels` as the first step in constructing the `index` for my `Series`, but you may do it however you see fit.

`extract_fig_1_frame`: This function takes the `DataFrame` you created in hw5, which looks like this:

	0101	0102	0103	0104
1979	14.7910	14.9970	14.9595	14.9220
1980	14.2000	14.2510	14.3020	14.3580
1981	14.2560	14.3560	14.4560	14.4455
1982	14.3515	14.4790	14.5605	14.6420
1983	14.2530	14.2795	14.3060	14.4000
1984	14.0050	14.1030	14.1700	14.2370

It contains data for each day of the year from 1979 through 2018. Return a `DataFrame` that looks like this:

	0101	0102	0103	0104	0105
mean	13.648313	13.713463	13.753975	13.798850	13.842487
two_s	1.117684	1.165589	1.183700	1.151764	1.141506

The `mean` row contains the mean of the data in a given column, the `two_s` row contains 2 × the standard deviation of the column.

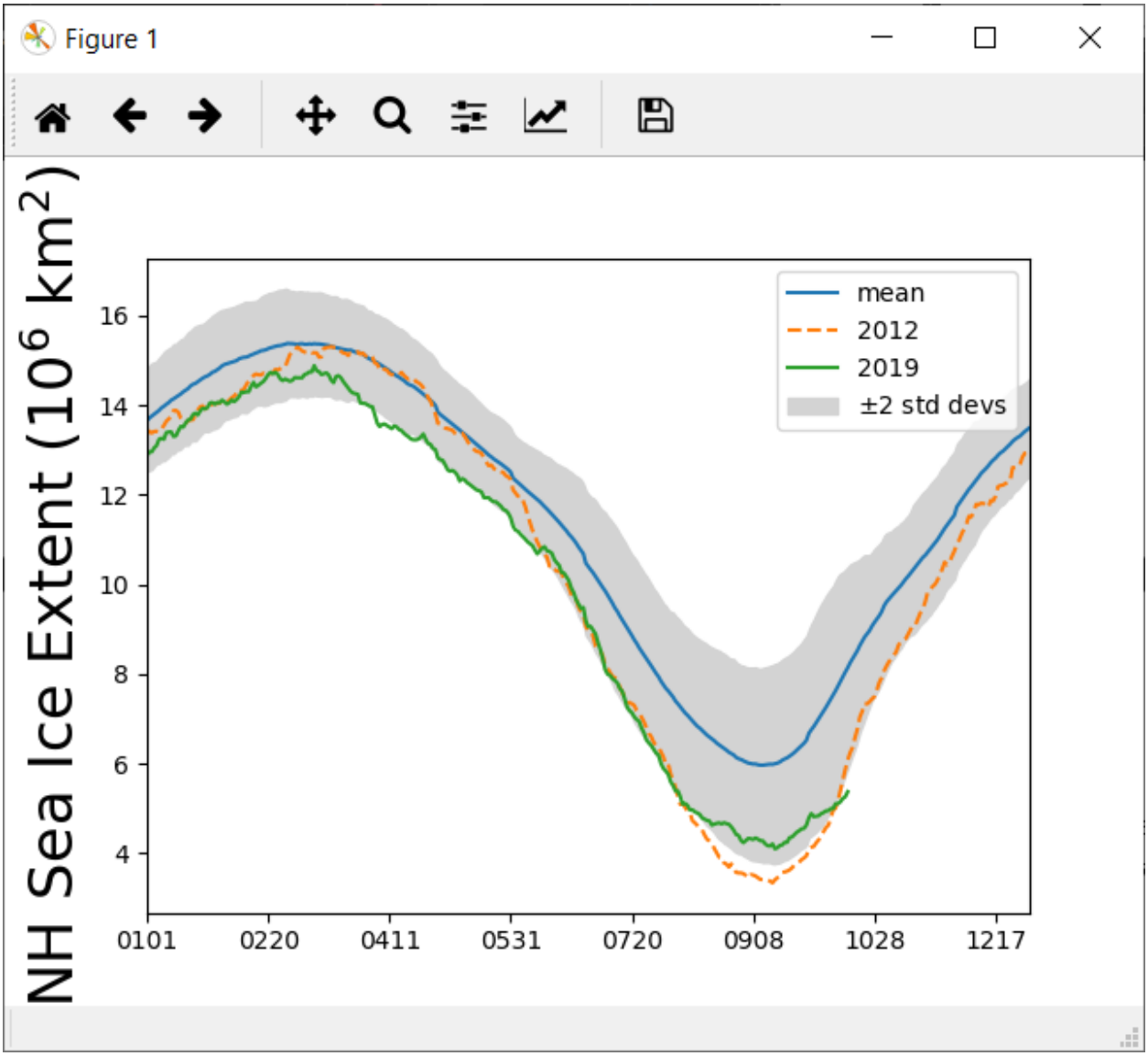
`extract_fig_2_frame`: This function takes the `DataFrame` you created in hw7 (illustrated at the bottom of the previous page) and returns a frame that looks like this:

	0101	0102	0103	0104	0105
1980s	14.1706	14.2604	14.3373	14.3964	14.4436
1990s	13.9606	14.0418	14.0667	14.0807	14.1123
2000s	13.3485	13.3712	13.4223	13.4738	13.5293
2010s	12.9272	12.9786	12.9929	13.0581	13.1024

The values are the decadal means for that date. The 2010's values are the means for the years 2010-2018.

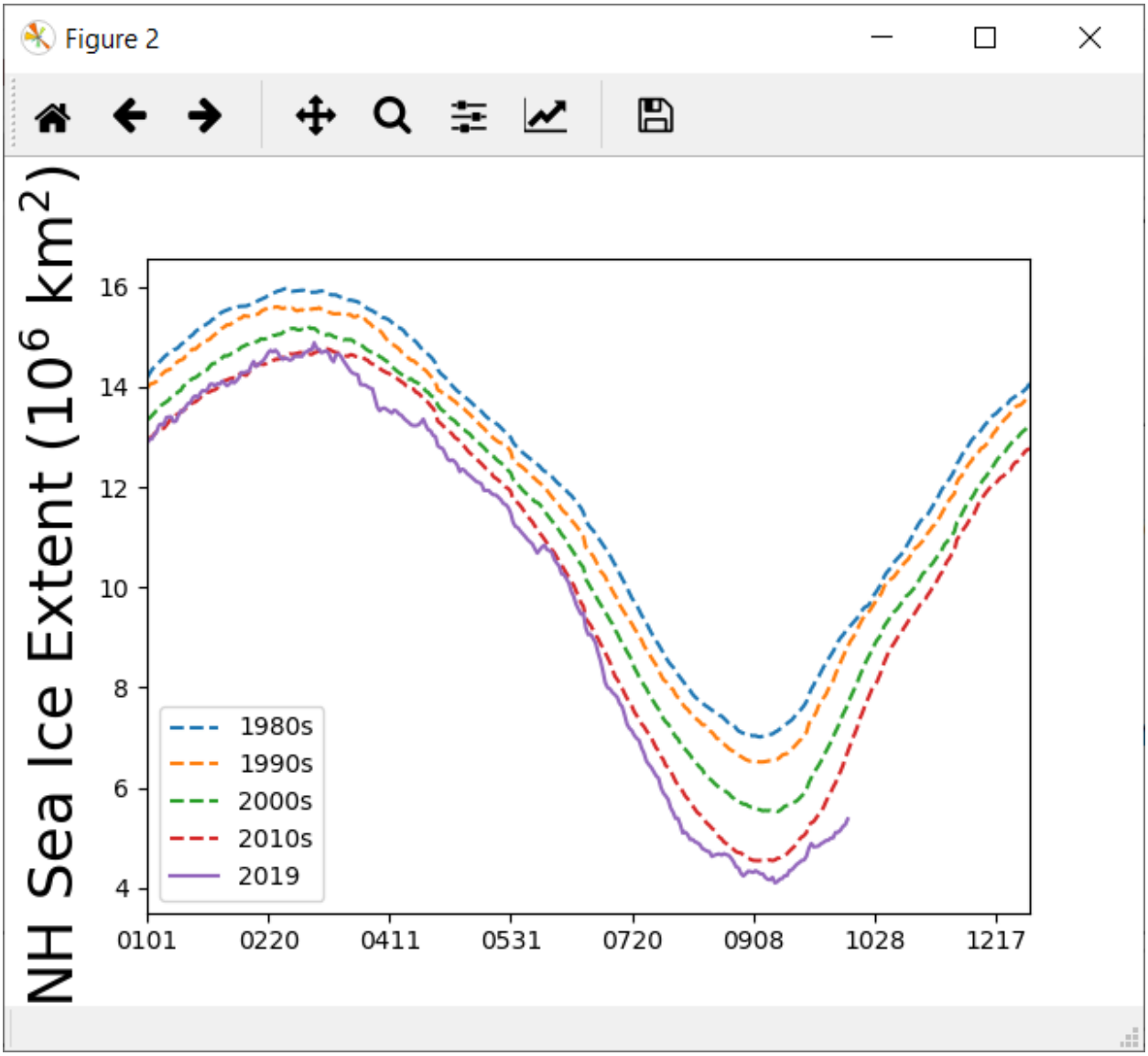
`make_fig_1`: This function takes a figure 1 frame and a hw5 frame, and creates a figure that looks like the image on the following page. The rubric for this figure:

- +4: the gray area looks like the image. It must not be bordered with visible lines.
- +2 each: the three curves look like the image.
- +4: the y-axis title looks like the image. Superscripts required for any title points.
- +2: correct x-axis tick labels.
- +4: legend looks like the image. Plus/minus symbol required for any legend points.



`make_fig_2`: This function takes a figure 2 frame and creates a figure that looks like the image on the following page. The rubric for this figure:

- +2 each: the five curves look like the image.
- +4: the y-axis title looks like the image. Superscripts required for any title points.
- +2: correct x-axis tick labels.
- +4: legend looks like the image.



`main`: Read the data in the file `data_79_18.csv` into a frame, recreating the hw7 frame. Make a figure 1 frame and a figure 2 frame. Make the figures (don't forget to call `plt.figure()` in between) and call `plt.show()` at the end.