In-class assignment #10

CMSE 402, Data Visualization Principles and Techniques
Spring 2022

Purpose: The purpose of this assignment is to for you to get some additional experience working with time series datasets, including data manipulation and visualization.

Part 1: Use the matplotlib `fill_between()` method to make a simple plot from the following arrays:

```
x = np.arange(0.0, 4.0*np.pi,0.01)
y1 = 2.0*np.sin(x)
y2 = y1-0.2-0.5*np.random.rand(x.size)
y3 = y1+0.2+0.5*np.random.rand(x.size)
```

where you draw `y1(x)` with a thick blue-dashed line, and the region between `y2` and `y3` with a semi-transparent red region. Use the zorder argument to ensure that the blue line is on top, and the `alpha` parameter to control the opacity of the region (note: to do this you may need to use the keyword arguments, or `kwargs`, for `fill_between()` – see this Stack Exchange question for useful information on how to do that).

Part 2: Now, apply these techniques to the temperature dataset that you used in the pre-class assignment. Create a plot with the following components for the city of your choice, using temperatures *in Celsius* (not Kelvin, as in the raw dataset):

1. The entire raw dataset as a function of time using a black translucent line, plotted as the bottom layer (i.e., the rear-most layer).
2. A filled red region that includes the entire temperature range for a given month (i.e., from the minimum temperature in that month to the maximum temperature in that month), as the next layer.
3. The mean monthly temperature (i.e., the average temperature for a given month), plotted as the top layer with a thick, blue solid line

Use NumPy array logic, array slicing, and array methods to get all of the required information out of your datasets – do not do it using loops if at all possible! Make sure to label your axes and give the plot a title. **If you have additional time,** (1) tweak the axis labels so that they look nice, and try to zoom in on individual years of data. Then (2) try to make a box-and-whisker plot (also see box plot) for a year of the data, showing individual months.

Hints: You may find it useful to use the python `datetime` module, and in particular the `strptime()` method, to extract the date information into a more usable format (although you DO have to figure out how to tell Python the format of the string that contains the date and time). Pandas has its own version of this, called `to_datetime()`, which is usable with Pandas data frames. When slicing numpy arrays, you may also find the NumPy `isfinite()` method and the Boolean logic functions useful for obtaining values in particular ranges and removing numbers.