STAT 29000 Project 11

Topics: Python, plotting

Motivation: The ability to take some data and produce a plot or graphic is a key skill for any scientist or researcher. In addition, although taking the time to thoroughly learn how to use a tool is most often the better thing to do, sometimes this is not practical and you will need to learn enough to "scrape by" and quickly do something.

Context: We just briefly reviewed some SQL, and learned the basics of making queries from within python. Now we will explore a variety of plotting package in python.

Scope: In this project you will exercise the ability to lightly modify or wrangle a dataset, quickly learn enough about a plotting library to produce a desired graphic, and rinse and repeat.

First, we must install some packages. In scholar, open up a shell and type the following:

```
python3.6 -m pip install plotnine --user
python3.6 -m pip install plotly --user
python3.6 -m pip install seaborn --user
python3.6 -m pip install bokeh --user
```

For the following questions, please copy and paste the following code at the top of your Jupyter notebook. We are simply importing useful Python modules that are required for this project. Be sure to replace "PURDUEALIAS" with your Purdue username. Note that if you get an error after running this chunk of code, you may need to click on Kernel->Restart.

```python
# the following two lines tell notebook.scholar.rcac.purdue.edu's default python interpreter
# that it should look in ~/.local/lib/python3.6/site-packages for packages as well
# if you do not include these two lines at the very top of your notebook, you won't
# be able to import and use the packages we installed at earlier times/dates
import sys
sys.path.append("/home/PURDUEALIAS/.local/lib/python3.6/site-packages")
```

After each problem, we've provided you with a list of keywords. These keywords could be package names, functions, or important terms that will point you in the right direction when trying to solve the problem, and give you accurate terminology that you can further look into online. You are not required to utilize all the given keywords. You will receive full points as long as your code gives the correct result.

Question 1: matplotlib

`matplotlib` could be considered the bread and butter of plotting in python.

Load the data here into a `pandas` dataframe. As you can see there is some identifying information, and a series of numbers that show how many confirmed cases of COVID-19 there are on a certain date.

Important note: It is not necessary for you to follow the hints in (1a) or (1b), however, they may be helpful as they are designed to walk you through what to do step-by-step.

1a. *(1 pt)* Create a standard (default options) area chart using `matplotlib` for the row of data where `Country/Region` is "Italy". Specifically, plot the number of confirmed cases (y-axis) by date (x-axis) from earliest to latest. Then use the code below to remove the x-axis labels.

```python
import matplotlib.pyplot as plt

# Code to create area chart

# Remove x-axis labels
plot = plt.gca()
plot.axes.get_xaxis().set_visible(False)
```

Hint: *To convert your row of data from **wide** to **long** format, use the `melt` function from the `pandas` library. Specifically, with the following options: `id_vars=["Province/State", "Country/Region", "Lat", "Long"], var_name="Date", value_name="Confirmed"`. This will make your dataframe look similar to below:*

```
   Province/State Country/Region   Lat  Long     Date  Confirmed
0             NaN          Italy  43.0  12.0  1/22/20          0
1             NaN          Italy  43.0  12.0  1/23/20          0
2             NaN          Italy  43.0  12.0  1/24/20          0
3             NaN          Italy  43.0  12.0  1/25/20          0
4             NaN          Italy  43.0  12.0  1/26/20          0
5             NaN          Italy  43.0  12.0  1/27/20          0
6             NaN          Italy  43.0  12.0  1/28/20          0
7             NaN          Italy  43.0  12.0  1/29/20          0
8             NaN          Italy  43.0  12.0  1/30/20          0
9             NaN          Italy  43.0  12.0  1/31/20          2
10            NaN          Italy  43.0  12.0   2/1/20          2
...
```

Keywords: *pandas: melt, loc; matplotlib: fill_between*

Item(s) to submit:

- A cell in the Jupyter notebook containing the python code used to create the graphic.

1b. *(2 pts)* There is some more data available:

Confirmed

Deaths

Recovered

Create a stacked area chart to plot all of this data (for Italy). We want area under the chart to show total confirmed cases where a portion of the cases resulted in recovery, and a portion resulted in death. For this reason, make sure to subtract the deaths and recoveries from the confirmed (as we will assume confirmed cases include those that resulted in either recovery or death). Remove the x-axis labels, and set the legend, using the code below:

```python
import matplotlib.pyplot as plt

# Code to create stacked area chart

# Remove x-axis labels
plt.legend(loc='upper left')
plot = plt.gca()
plot.axes.get_xaxis().set_visible(False)
```

Hint: *If you get an all black chart, in your `p9.aes(...)` part, make sure you use `fill="Category"` not `group="Category"`.*

Some ordered hints:

1. Read in the datasets using `pandas read_csv`.
2. Add a `Category` column to each dataset where each row is a repeated value of: "Confirmed", "Deaths", or "Recovered", depending on which dataset you are working with.
3. Combine the dataframes using `pandas append` function with the argument `ignore_index=True`.
4. Isolate the "Italy" data (should be 3 rows, one from each `Category`.
5. Use the `pandas melt` function with the following options: `id_vars=["Province/State", "Country/Region", "Lat", "Long", "Category"], var_name="Date", value_name="Value"` to convert from wide to long.
6. Use `pandas to_datetime` to convert the `Date` column to the proper type.
7. Use `pandas pivot` function to convert the data back out to wide format where there are three columns: `Confirmed`, `Deaths`, and `Recovered` which each row has a date and a numeric value. *Hint:* If you've followed these hints, you can use the following arguments to `pivot`: `index="Date", columns="Category", values="Value"`.
8. Subtract both the number of deaths and number of recovered from the number of confirmed.
9. Use `stackplot` to create the plot. *Hint:* The x-axis *date* information may be accessed via `mydataframe.index`.

Keywords: *pandas: append, loc, melt, to_datetime, pivot; matplotlib: stackplot*

Item(s) to submit:
- A cell in the Jupyter notebook containing the python code used to create the graphic.

Question 2: plotnine

`plotnine` is a python package that aims to follow the Grammar of Graphics, and is based on `ggplot2` from R.

2a. *(1 pt)* Repeat (1a) using the `plotnine` library.

Hint: *Here are a list of the functions/objects you will need from* `plotnine`*:*

- p9.ggplot
- p9.aes
- p9.geom_area

Hint:

```
import plotnine as p9

(
# Code to create area chart


# to remove the x-axis labels, "add" this to your plot
+ p9.theme(axis_title_x=p9.element_blank(),
        axis_text_x=p9.element_blank())
)
```

Item(s) to submit:
- A cell in the Jupyter notebook containing the python code used to create the graphic.

2b. *(1 pt)* Repeat (1b) using the `plotnine` library.

Hint: *The only modification you need to make to the plot code is to add* `fill='group'` *to the* `p9.aes` *function, where 'group' is a column of a dataframe with the three categories:* `confirmed`*,* `dead`*, and* `recovered`*. See a snippet of the dataframe below:'*

```
100 2020-03-07     Deaths    233
155 2020-03-07  Recovered    589
45  2020-03-07  Confirmed   5061
156 2020-03-08  Recovered    622
46  2020-03-08  Confirmed   6387
101 2020-03-08     Deaths    366
```

Hint: *Assuming* `italy` *is your "melted" dataframe from (2a), the following code will put your dataframe in the proper format so it is ready to plot.*

```
# convert date to date
italy['Date']= pd.to_datetime(italy['Date'])

# convert back out to wide a bit
italy = italy.pivot(index="Date", columns="Category", values="Value")

# remove deaths and recovered from confirmed
italy["Confirmed"] = italy["Confirmed"] - italy["Deaths"]
italy["Confirmed"] = italy["Confirmed"] - italy["Recovered"]

# go back to longer
italy = italy.reset_index()
italy = pd.melt(italy, id_vars="Date")
```

Item(s) to submit:

- A cell in the Jupyter notebook containing the python code used to create the graphic.

Question 3: plotly

`plotly` is a library that aims to create *interactive* graphics (graphics that you can interact with, click, hover, etc.). It has good Jupyter notebook integration and is therefor particularly useful when the end user will be looking at the graphics within the notebook (rather than publishing in an article, for example).

(1 pt)

Read in the data from here. Use the `plotly` library to create a dot plot with the country name on the y-axis and count on the x-axis. Make the title of the plot "Gun Murders". Make the x-axis title, "Gun murders (per 100,000)". Make the y-axis title, "Country".

Keywords: *plotly.express.scatter*

Item(s) to submit:

- A cell in the Jupyter notebook containing the python code used to create the graphic.

Question 4: seaborn

`seaborn` aims to be a high-level interface for graphics built on `matplotlib`. It is a decent solution when you just want easily accessible statistical plots.

(1 pt)

Read in the data from here. Use the `seaborn` library to create a boxplot where `revenue_category` is shown on the x-axis and `revenue` is shown on the y-axis. Rotate x-axis labels 90 degrees.

Keywords: *boxplot, set_xticklabels*

Item(s) to submit:
- A cell in the Jupyter notebook containing the python code used to create the graphic.

Question 5: bokeh

`bokeh` is an interactive visualization library (like `plotly`) for modern web browsers. It's Jupyter notebook integration is not as good as `plotly` (as you will see), but it is easy to make some very attractive, interactive visualizations.

(2 pts)

Read in the data from here. Group the data by `playlist_genre` and get the average `track_popularity`. Use the `bokeh` library to create a barplot showing the average track popularity by `playlist_genre`.

Keywords: *pandas: groupby, mean, unique; bokeh: figure, vbar, show*

Item(s) to submit:
- A cell in the Jupyter notebook containing the python code used to create the graphic.

Question 6: open

(1 pt)

Read in a new dataset (or dataset previously provided), and create any cool graphic you want using your favorite plotting library (you are not limited to the 5 libraries used in this project). Use as many features of the library as you'd like – the graphic can be as simple or complicated as you'd like, but must be *new*.

Item(s) to submit:
- A cell in the Jupyter notebook containing the python code used to create the graphic.

Project Submission:

Submit your solutions for the project at this URL: https://classroom.github.com/a/59lzU-Tm using the instructions found in the GitHub Classroom instructions folder on Blackboard.

Important note: Make sure you submit your solutions in both .ipynb and .pdf formats. We've updated our instructions to include multiple ways to convert your .ipynb file to a .pdf on scholar. You can find a copy of the instructions on scholar as well: `/class/datamine/data/spring2020/jupyter.pdf`. If for some reason the script does not work, just submit the .ipynb.