Homework 10: In-class Activity

December 19 - November 29

1 Plotting Data

In this activity, you’ll be learning to visualize your data using the very powerful plotting library Plotly. This hands-on experience will be useful practice for implementing your final project. Nothing needs to be officially submitted for this activity; instead, in order to receive full credit, you need to show a TA each of your final graphs. Working with a partner is allowed and encouraged.

2 Task 1: Getting Started

2.1 Installation

To get set up, we’ll first need to install the Plotly library. You should be able to do this by opening up your terminal or command line and typing:

`conda install plotly`

Conda is a Python package manager included with Anaconda. It helps find Python modules that we don’t currently have installed on our computer and installs them. This command should show a bunch of messages, including a list new packages to install and additional packages to update. It will ask you if you wish to confirm the installation. Press enter to let Anaconda install these packages. This should print out a bunch of messages, ending with something along the lines of Linking packages COMPLETE. If you instead see an error message or red text, please call over a TA.

First, you will need to upgrade your version of plotly to the latest version. Run the command:

`pip3 install plotly --upgrade`

If it complains about pip3 not being a valid command, then use pip.

In the latter tasks, if your program errors due to not having the geopandas or pyshp libraries, try installing them with the following commands in the terminal:

```
conda install -c conda-forge geopandas
conda install -c conda-forge pyshp
```

2.2 Setup

**Action:** For this in-class activity, download `example_graphs.py` from the course website.

This shows example graphs which you can alter to perform the tasks below.
3 Task 2: Bar Graph

In `example_graphs.py`, we provide a Bar Graph example. First, notice the import statements at the top of the file. They allow us to use plotly. Namely, the “graphic object” and ability to produce offline (i.e., non-interactive, non-online examples).

**Action:** First, run this code to ensure it works on your computer and that the packages are properly installed.

3.1 Defining Data

Next, we see that the code creates a graph object (named `data`) for the bars themselves. `go.Bar` takes in the x and y values as parameters. The x-axis is always the horizontal axis, and the y-axis is the vertical axis. These two parameters are mandatory; as it wouldn’t make sense to have a bar graph without this minimal amount of information. Notice that the x and y parameters are written in the same notation as a dictionary, where ‘x’ is one of our keys, and its corresponding list of `['Cat', 'Dog']` is its value. Likewise, ‘y’ has its own list `[4, 11]` which corresponds to Cat and Dog, respectively.

3.2 Layout

Next, the `Layout` object allows us specify the visual properties of the graph at large, such as the titles.

3.3 Bringing it all together

We need to construct an overall Figure, which at a minimum needs data to be specified (passed in as a parameter), and optionally Layout properties to be specified (passed in as a parameter). We have already defined both of those in our two lines above, so Line 11 just constructs a Figure from them.

**Action:** Run your code (only the Bar Graph example should be executed, so comment out all other code examples). It should produce a simple bar graph. If it does not, try to debug it and ask a TA for help if need be.

4 Task 3: Making your Own Bar Graph

**Action:** Download our small movies dataset from the course website (`movies.csv`).

We will use Pandas to read it in. The file is actually delimited by semi-colons, not commas. No worries; Pandas has your back.

**Action:** Parse the file into a data frame via:

```python
df = pd.read_csv('movies.csv', sep=';')
```
For this task, we want you to plot the total gross earning corresponding to each director, across all decades. That is, each director will be represented by just 1 bar on the x-axis, and the height of the bar will be designated by his/her movies’ total gross earnings.

So, for each row (i.e., director), we need to sum up all of the columns. Pandas can sum the columns for us, via:

```python
def['gross'] = df.sum(axis=1)
```

That is, we are defining a new column named “gross” within our data frame “df”, and each of its values is merely the sum of the columns. Print out this data frame’s series. Who has the highest value? (the e+N is scientific notation, denoting the first number should be multiplied by 10^N). Notice how it could be a bit slow having to look at each individual value, and that textual representation isn’t the easiest. Hence, why we should visualize it!

To make your own graph, we first need to replace data that comprises the Bar graph. We still need to tell our graph to use ‘x’ and ‘y’ keys, but the values for each need to be replaced by our data frame columns corresponding to “directors” names and “gross” earnings, respectively. We should also appropriately change our titles and axis values, as we are no longer concerning dogs and cats.

Action: Now, try to plot the graph (it should contain 8 bars, with Spike Lee being the shortest)!

### 5 Task 4: Change the Bar Colors

Inevitably, any graph you care to plot on your own will involve making small changes to the canonical examples provided at plot.ly/python. So, you’ll need to learn how to use and edit their code. For practice with such, let’s make a small change to our graph by changing the colors of the bars.

Visit https://plot.ly/python/bar-charts/ and scroll down to the “Bar Chart with Hover Text” example. Notice how that graph uses different colors – it’s a cool, lightly shaded baby blue.

Action: Make our graph’s bars have the same color. We guide you below.

Looking at their provided code, we see a property called “color” so that should indicate to us that that’s where the color is being defined. Specifically, it is part of a section that is defining properties for a “marker”. We don’t know what “marker” stuff is, but if we could just copy and paste that stuff into our code, then it will allow us to experiment with it. So, we must figure out how to even paste that “marker” section into our code in a way that makes sense.

If we look just 2-3 lines above that, we see our familiar ‘x’ and ‘y’! Sweet. We see that this example uses ‘x’ and ‘y’ a little differently than how we do. They specify a list of properties within the construction of Bar, but we are doing so via defining everything with the syntax a dictionary. So, instead of doing things like x=[list data here] like they do, our code performs the same stuff by ‘x’: [list data here]. Thus, if we want to define a “marker”, we shouldn’t write marker = dict( ... ) like they do, but just slightly change it to being 'marker': dict( ... ). The same is true for the opacity property. Instead of opacity=0.6, we should write ‘opacity’: 0.6. To be clear, our go.Bar object should have 4 properties defined in the syntax that looks like a dictionary: x, y, marker, and opacity (each of which are keys).
Action: Plot it! It should have baby blue bars now.

6 Task 5: Geographic Plot

We will now use the geographic example code that we provide in example_graphs.py. So, comment out all code except for this example.

Action: Run the provided example, first, just to ensure it works for you (ensuring you have all necessary packages installed). It should illustrate a few counties in Southern California as having the highest unemployment.

Now, instead of plotting unemployment data, let’s plot election data.

Action: Ensure you’ve downloaded the election-context-2018.csv file from our course website.

Note, we already import the data for you into a data frame called data_frame. A choropleth graph is created similar to a Bar Graph in that we need to specify the data (location data) and its values. Then, the rest is all about different display properties like titles, colors, etc. In the example we provide, we are plotting data on a per-fips basis, as opposed to per-state or per-country. Luckily, for the election data, we will be doing the same thing – plotting data on a per-fips basis. However, instead of the fips data coming from

```python
fips = df_sample['FIPS'].tolist()
```

we need to create a different variable which uses the fips data from our election’s data_frame.

Action: Change our fips data to use our data_frame’s fip info.

Next, we just need to change our actual values. In the example we provided, we were using data from the df_sample’s column named “Unemployment Rate (%)

```python
values = df_sample['Unemployment Rate (%)'].tolist()
```

Instead, let’s use the # of votes that were for Hillary Clinton in 2016.

Action: Change the code so that instead of values being equal to the aforementioned df_sample column, make it use the number of votes for Hillary in 2016. Plot the graph. Does it actually work?

Action: Change the titles so that the graph actually makes sense and isn’t claiming to be about unemployment data.

Action: Now, instead of being the raw votes for Hillary, let’s plot the % of votes that were for Hillary. That is, our values should be #_for_hillary / (#_for_hillary + #_for_trump)

Action: Hover your mouse over the county that has the highest % of votes for Hillary. Which county is this?
7 Hand-in

You do not need to submit this assignment to Canvas. Instead, the goal is for you to complete this assignment in-class (we aim to provide you with enough in-class time to do so). You just need to show a TA your final graph. We encourage you to work with a partner, which is fruitful practice for working on your final project.