“We global”
-- Professor DJ Khaled
Announcements

• Final Project Proposals:
  • List who your partner is (if applicable), and have only 1 person submit the proposal
  • Each person will receive the same grade
• Think of what you want me to cover for the final lecture(s)
• Final Project MUST NOT BE LATE – Dec 3.
• My new office: CIT 317
Visualizing Data -- Outline

• Different Types of Geographic Graphs
• Plotly Examples
• Real-Time Coding / Practice
Lately, we’ve used incredibly useful external libraries (e.g., Pandas), why not make 100% of the class revolve around using these?
Lately, we’ve used incredibly useful external libraries (e.g., Pandas), why not make 100% of the class revolve around using these?

• We needed a foundation in computing first, allow us:
  • To understand how to use the libraries
  • Do anything we want – not confined to others’ libraries
  • Libraries can come and go and change, but a diamond foundation is forever.
Choropleth graph

- colors distinct regions according to the information being visualized
- choropleth graphs can plot data using a color scale
Scatter map

- Plots a point for every 25 people

- Easily visualize where there is population density of each race

- This type of map by Radical Cartography has brought about discussion regarding segregation in modern cities
Bubble Map

The geography of bowling (larger circles = more bowling centers)

<table>
<thead>
<tr>
<th># of bowling centers</th>
<th>Centers per capita</th>
</tr>
</thead>
</table>

Map of the United States with bubble map showing the distribution of bowling centers.
## these are your import statements

```python
import pandas as pd
from plotly.offline import plot
import plotly.graph_objs as go
```

## you can also import csv from Kaggle or from your own personal directory

```python
data_frame = 
```
```python
data_frame.head()
```

<table>
<thead>
<tr>
<th>code</th>
<th>state</th>
<th>category</th>
<th>total exports</th>
<th>beef</th>
<th>veggies</th>
<th>proc</th>
<th>total veggies</th>
<th>corn</th>
<th>wheat</th>
<th>cotton</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL</td>
<td>Alabama</td>
<td>state</td>
<td>1390.63</td>
<td>34.4</td>
<td>...</td>
<td>8.9</td>
<td>14.33</td>
<td>34.9</td>
<td>70.0</td>
<td>317.61</td>
</tr>
<tr>
<td>AK</td>
<td>Alaska</td>
<td>state</td>
<td>13.31</td>
<td>0.2</td>
<td>...</td>
<td>1.0</td>
<td>1.56</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>AZ</td>
<td>Arizona</td>
<td>state</td>
<td>1463.17</td>
<td>71.3</td>
<td>...</td>
<td>239.4</td>
<td>386.91</td>
<td>7.3</td>
<td>48.7</td>
<td>423.95</td>
</tr>
<tr>
<td>AR</td>
<td>Arkansas</td>
<td>state</td>
<td>3586.02</td>
<td>53.2</td>
<td>...</td>
<td>7.1</td>
<td>11.45</td>
<td>69.5</td>
<td>114.5</td>
<td>665.44</td>
</tr>
<tr>
<td>CA</td>
<td>California</td>
<td>state</td>
<td>16472.88</td>
<td>228.7</td>
<td>...</td>
<td>1303.5</td>
<td>2106.79</td>
<td>34.6</td>
<td>249.3</td>
<td>1064.95</td>
</tr>
</tbody>
</table>

[5 rows x 17 columns]
Creating your choropleth

```python
choropleth = go.Choropleth(z = data_frame['total exports'],
                         locations = data_frame['code'],
                         locationmode = 'USA-states')
```

*z* is the data you want to graph

*locations* is the corresponding column to that data *(z)*

*locationmode* can be either:

- `'ISO-3'` country codes (which is the default) – [check here for a listing](https://plot.ly/python/reference/#choropleth-locationmode)
- `'USA-states'`
- `'country names'`

## provides the parameters how the map will be displayed

```python
layout = go.Layout(title = 'Total exports of the US', 
    geo = {'scope': 'usa', 'projection': {'type': 'albers usa'}})
```

- title of the graph

- `geo` refers to geo-specific information, which is the form of a dictionary
  - `scope` is the region the map contains
  - `type` is the type of map
Drawing the graph

```python
define figure = go.Figure(data = [choropleth], layout = layout) plot(figure)
```
Customizing the Figure

```python
## color_scale = [[0.0, 'Blue'], [1.0, 'Red']]
color_scale = [[0.0, 'Blue'], [0.5, 'White'], [1.0, 'Red']]## add a new attribute (colorscale stuff) to the choropleth
choropleth = go.Choropleth(z = data_frame['total exports'],
                           locations = data_frame['code'],
                           locationmode = 'USA-states',
                           colorscale = color_scale,
                           autocolorscale = False)

## we turn off the autocolorscale by assigning it to False
## re-plot!
plot(figure)
```
Customizing the Figure

```python
## color_scale = [[0.0, 'Blue'], [1.0, 'Red']]
color_scale = [[0.0, 'Blue'], [0.5, 'White'], [1.0, 'Red']]

## add a new attribute (colorscale stuff) to the choropleth
choropleth = go.Choropleth(z = data_frame['total exports'],
                          locations = data_frame['code'], locationmode = 'USA-states',
                          colorscale = color_scale, autocolorscale = False)

## we turn off the autocolorscale by assigning it to False
## re-plot!
plot(figure)
```
Adding Text Labels

```python
text_labels = data_frame['state'].astype(str) + "</br>Total Exports: " + data_frame['total exports'].astype(str)
```

```python
## add text labels
choropleth = go.Choropleth(z = data_frame['total exports'],
locations = data_frame['code'], locationmode = 'USA-states',
colorscale = color_scale, autocolorscale = False, text = text_labels )

plot(figure)
```

`astype` is similar to `str()` type casting except it converts each column into a string

`</br>` is the HTML version of newline
Adding Text Labels

```python
text_labels = data_frame[‘state’].astype(str) + “</br>Total Exports: “ + data_frame[‘total exports’].astype(str)
## add text labels
choropleth = go.Choropleth(z = data_frame[‘total exports’],
locations = data_frame[‘code’], locationmode = ‘USA-states’,
colorscale = color_scale, autocolorscale = False, text = text_labels
plot(figure)
```

`astype` is similar to `str()` type casting except it converts each column into a string

`</br>` is the HTML version of newline
Final Result

Total exports of the US

- Wyoming: Total Exports: 349.69

- California

- Alabaik
We can do the same for other geographic maps!

```python
# make your data frame
# try with

scattergeo = go.Scattergeo(lat = data_frame['lat'], lon = data_frame['long'],
node = 'markers', locationmode = 'USA-states')

layout = go.Layout(title = “Map of US major airports”, geo = {'scope': ‘usa’,
‘projection’: {'type: ‘albers usa’}})
```
Should look something like this!
Adding Text Labels and Markers

text_labels = data_frame[‘airport’].astype(str) + “</br>Takeoff/Landings: “ + data_frame[‘cnt’].astype(str)

## if you would like this to be a bubble map rather than a uniform scatter map
scattergeo = go.Scattergeo(lat = data_frame[‘lat’], lon = data_frame[‘long’], mode = ‘markers’, locationmode = ‘USA-states’, text = text_labels, marker = {'size': data_frame[‘cnt’] / 100})

## we adjust this last attribute so it looks right. Something like:
## will make sure bubbles don’t get too small or too large
maker = {'size': 10 + data_frame[‘cnt’] / 500} #
Adding Text Labels and Markers

```python
# Adding text labels

text_labels = data_frame['airport'].astype(str) + "</br>Takeoff/Landings: " + data_frame['cnt'].astype(str)

## if you would like this to be a bubble map rather than a uniform scatter map

scattergeo = go.Scattergeo(lat = data_frame['lat'], lon = data_frame['long'], 
                           mode = 'markers', locationmode = 'USA-states', text = text_labels, 
                           marker = {'size': data_frame['cnt'] / 100})

## we adjust this last attribute so it looks right. Something like:
## will make sure bubbles don’t get too small or too large
maker = {'size': 10 + data_frame['cnt'] / 500} #
```
REAL-TIME CODING