# “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

# Warm-up

Lately, we've used incredibly useful external libraries (e.g., Pandas), why not make 100% of the class revolve around using these?

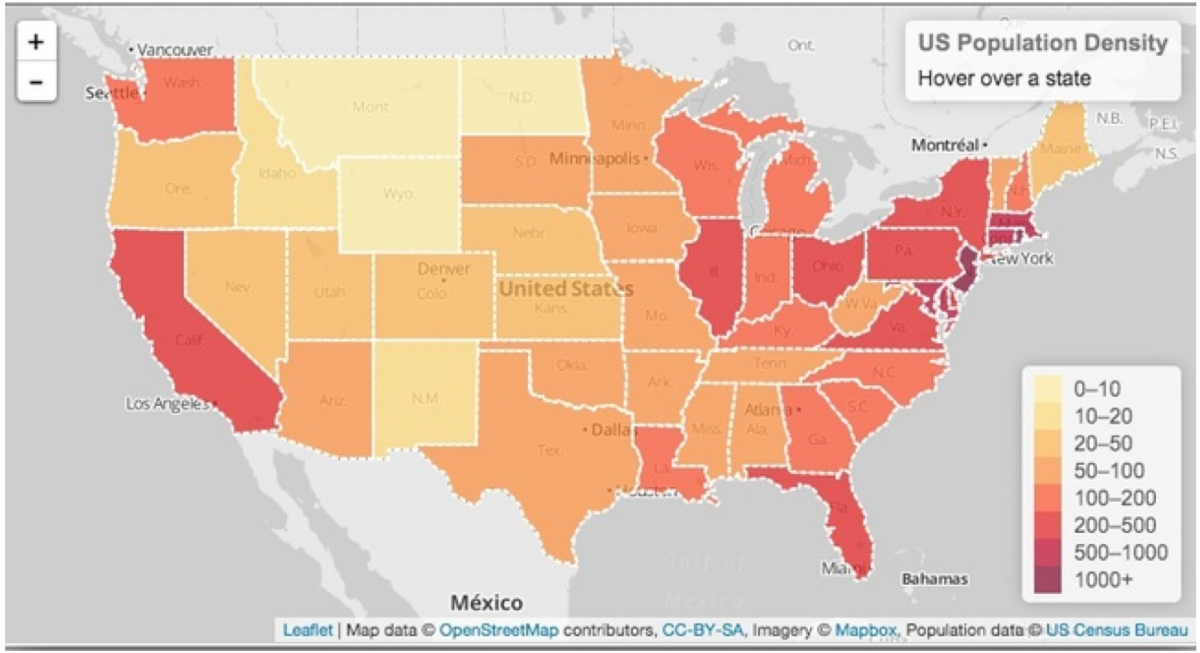
# Warm-up

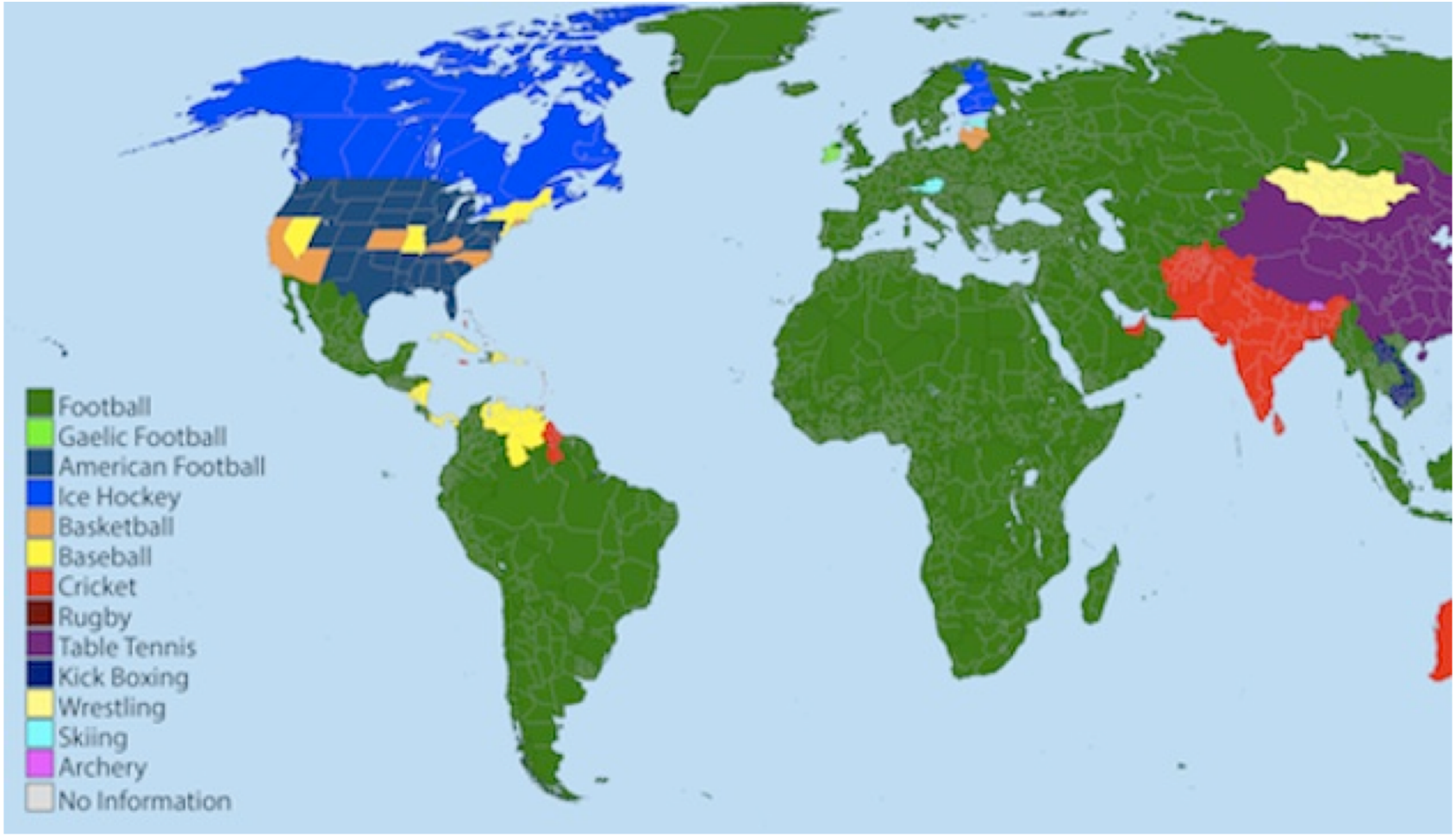
Lately, we've used incredibly useful external libraries (e.g., Pandas), why not make 100% of the class revolve around using these?

- 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

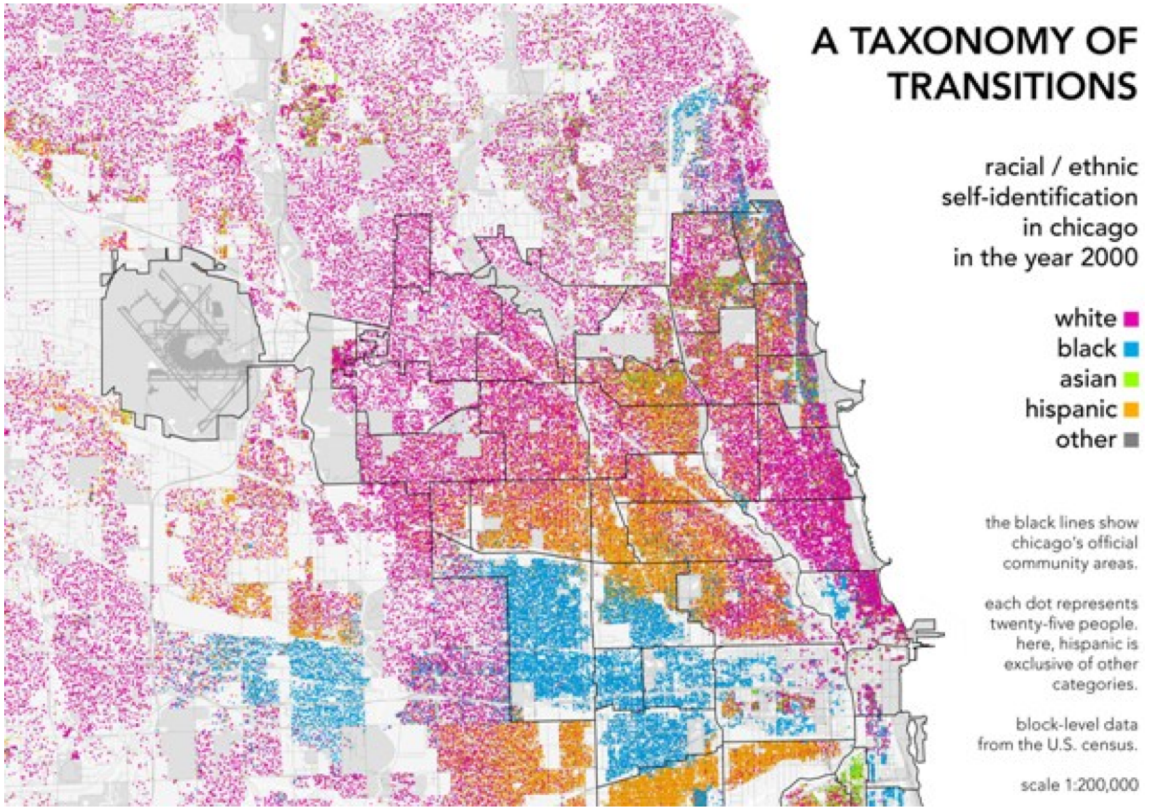
- colors distinct regions according to the information being visualized
- choropleth graphs can plot data using a color scale



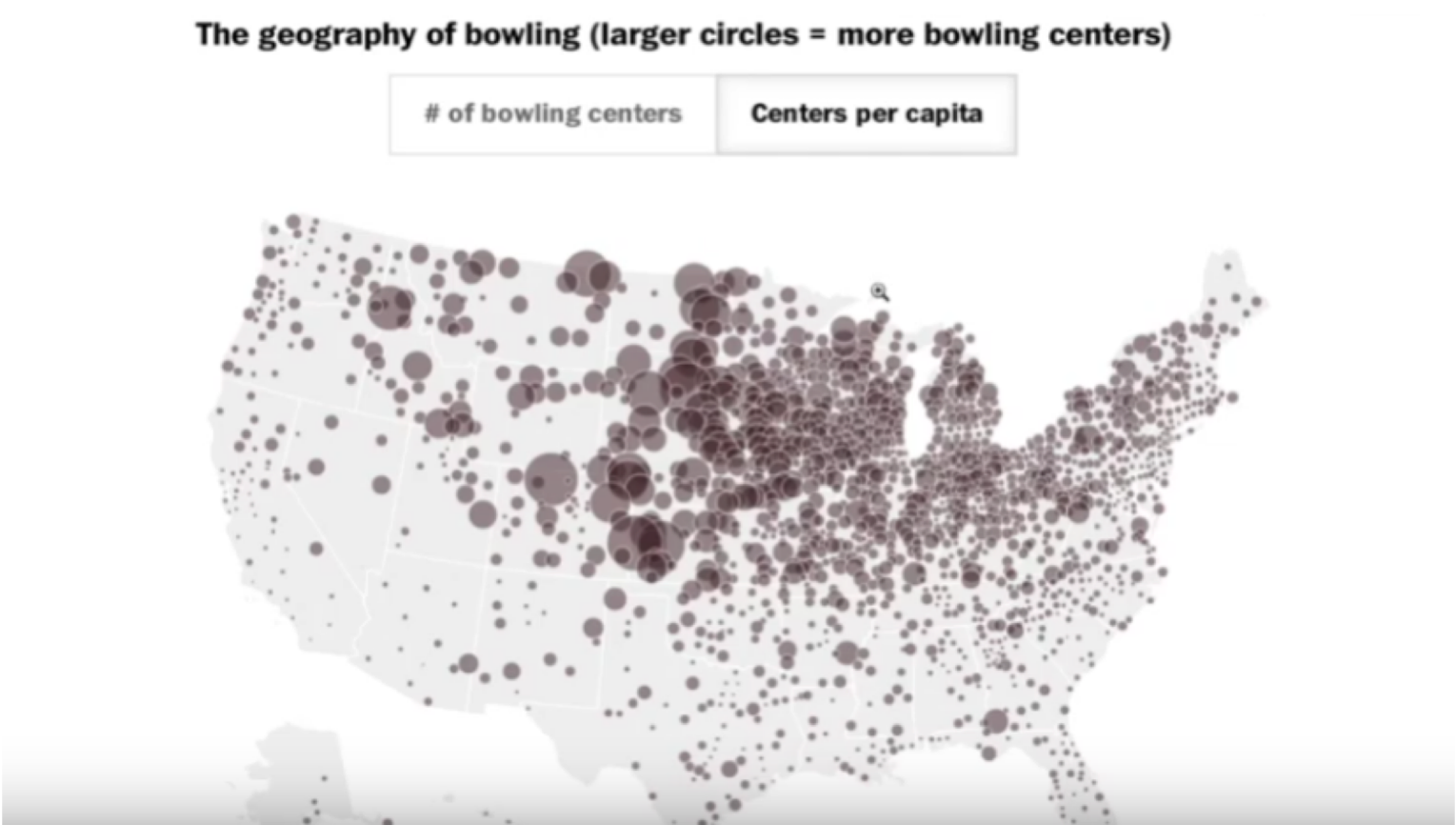

Football
Gaelic Football
American Football
Ice Hockey
Basketball
Baseball
Cricket
Rugby
Table Tennis
Kick Boxing
Wrestling
Skiing
Archery
No Information

# 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)
# of bowling centers
Centers per capita

# How to make this in Python: Getting Started

## these are your import statements

```
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

```
data_frame =
pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2011_us_ag_exports.csv')
```

# data_frame.head()

| | code | state | category | total exports | beef | ... | veggies proc | total veggies | corn | wheat | cotton |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | AL | Alabama | state | 1390.63 | 34.4 | ... | 8.9 | 14.33 | 34.9 | 70.0 | 317.61 |
| 1 | AK | Alaska | state | 13.31 | 0.2 | ... | 1.0 | 1.56 | 0.0 | 0.0 | 0.00 |
| 2 | AZ | Arizona | state | 1463.17 | 71.3 | ... | 239.4 | 386.91 | 7.3 | 48.7 | 423.95 |
| 3 | AR | Arkansas | state | 3586.02 | 53.2 | ... | 7.1 | 11.45 | 69.5 | 114.5 | 665.44 |
| 4 | CA | California | state | 16472.88 | 228.7 | ... | 1303.5 | 2106.79 | 34.6 | 249.3 | 1064.95 |

[5 rows x 17 columns]

# Creating your choropleth

```
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
- **'USA-states'**
- **'country names'**

https://plot.ly/python/reference/#choropleth-locationmode

# Layout

## provides the parameters how the map will be displayed

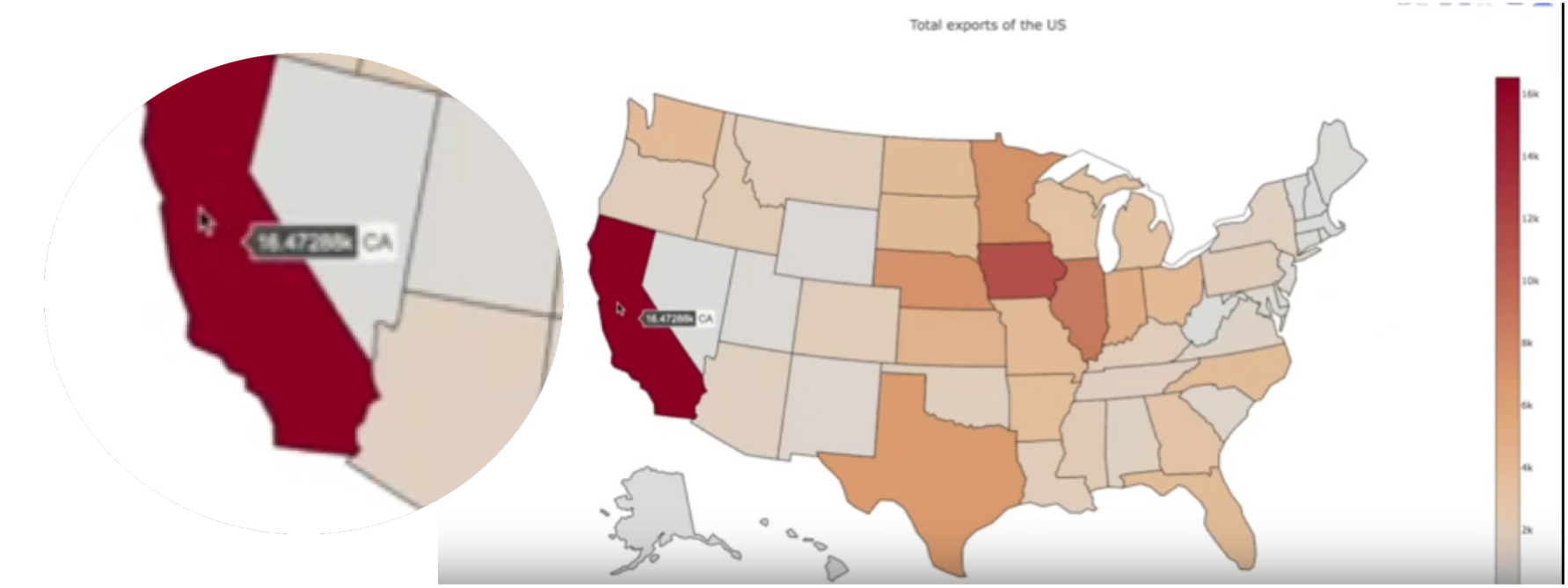
```
layout = go.Layout(title = 'Total exports of the US', \
    geo = {'scope': 'usa', 'projection': {'type': 'albers usa'}})
```

- title of the graph

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

# Drawing the graph

```
figure = go.Figure(data = [choropleth], layout = layout)
plot(figure)
```



# Customizing the Figure

```
## 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

```
## 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

```
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

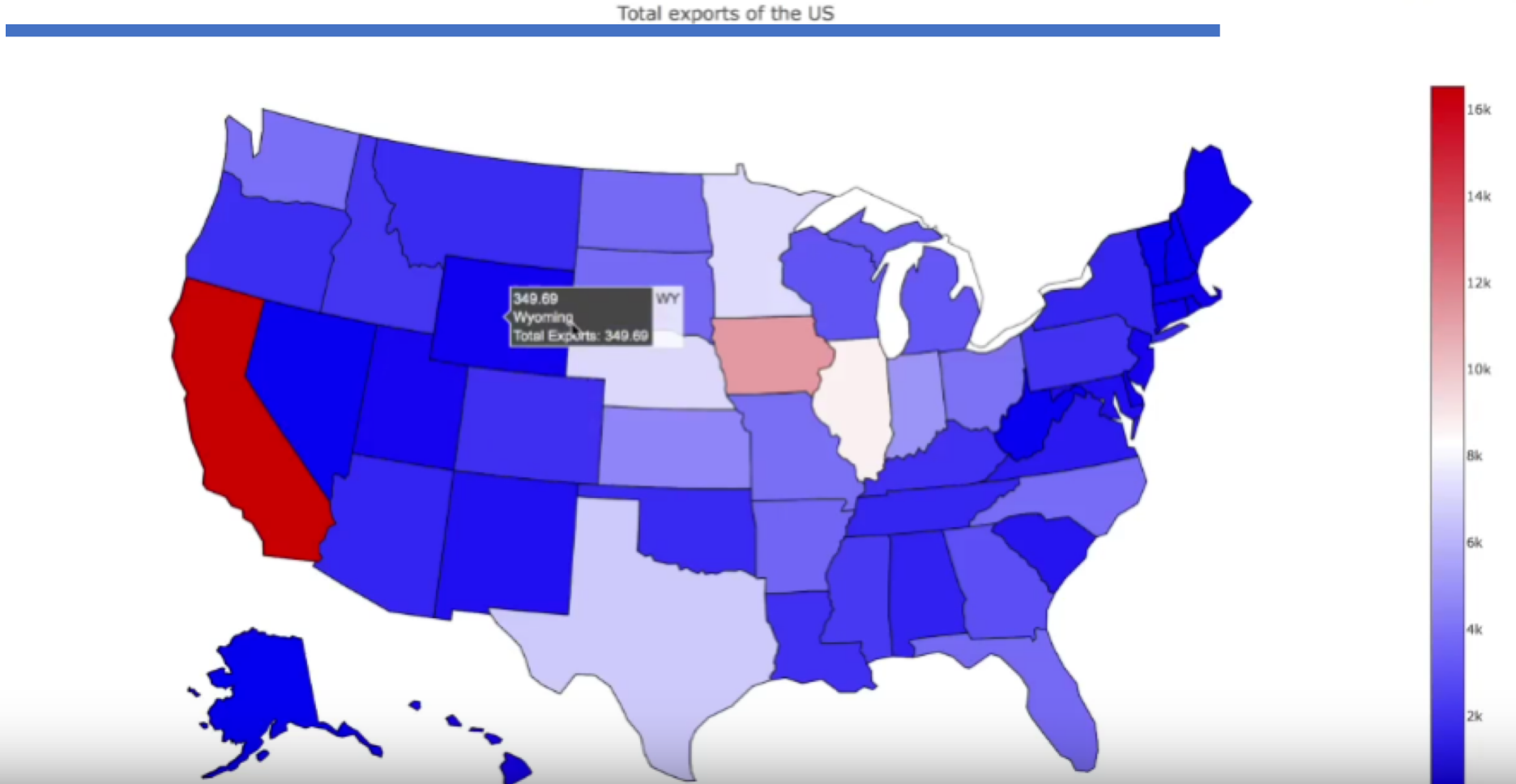
# Adding Text Labels

```
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



# We can do the same for other geographic maps!

**# make your data frame**
**# try with**
**http://raw.githubusercontent.com/plotly/datasets/master/2011_february_us_airport_traffic.csv'**

**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'}})

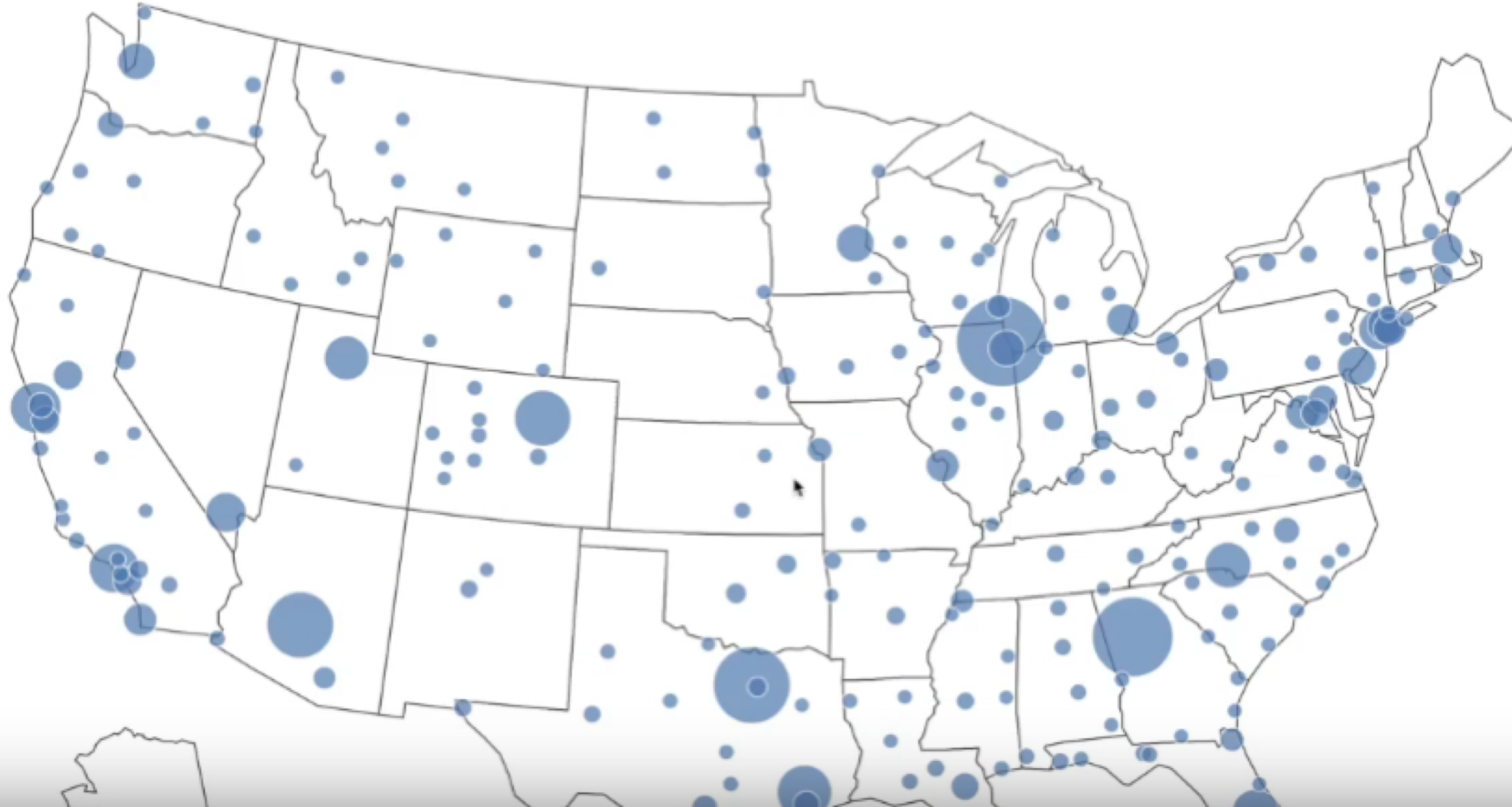
# Should look something like this!

# 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

```
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

# LAB TIME