YData: Introduction to Data Science

Class 15: Mapping

Overview

Very quick review of interactive graphics with plotly

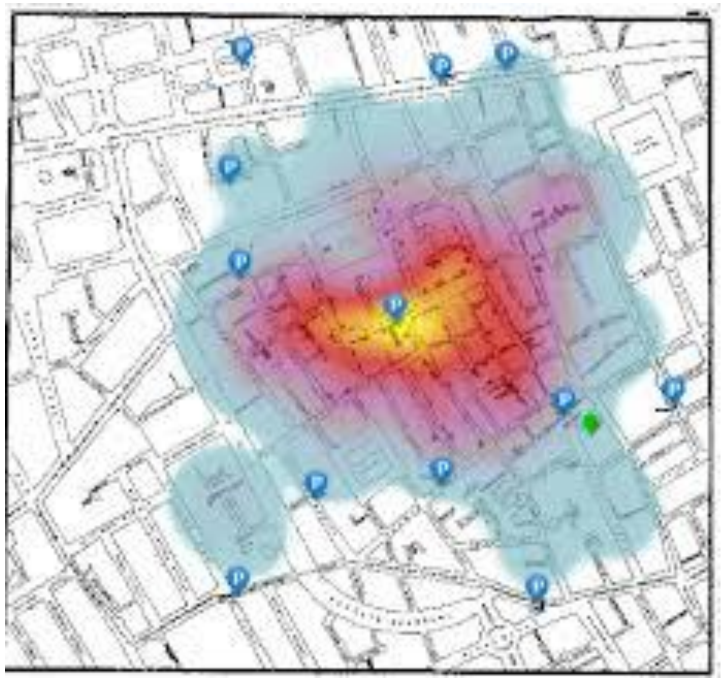
Heatmaps

Maps

- geopandas
- Coordinate reference systems and projections
- Choropleth maps

If there is time

- For loops



John Snow's ghost map

Reminder: class project

The class project is a 6-10 page Jupyter notebook report where you analyze data you find interesting

Think about what questions you want to examine, find data, and load it into Python

- A few sources for data sets are listed on Canvas

You can download a project template Jupyter notebook using:

```
import YData
YData.download_class_file('project_template.ipynb', 'homework')
```

A polished draft of the project is due on November 16th

Collaborative projects?

Note: you can submit a collaborative project with one other person

- Project should be twice as long and twice as impressive!
  - i.e., 10-16 pages, more in depth analyses, etc.

Homework 6 is due on Sunday

- We will cover all material you need to complete the homework after today's class, so start early so you have some time to also work on your project

Finally, I encourage everyone to continue to attend practice sessions

- Particularly if you found the midterm difficult

Where we are and where we're going…

What we have covered:

- What is Data Science
- Basics of Python (data types, lists, etc.)
- Numerical computations (numpy)
- Data tables (pandas)
- Data visualization (matplotlib and seaborn)
- Interactive graphics

Today: Mapping

The rest of the semester:

- Functions and for loops
- Statistical analysis
- Machine Learning
- Ethics and conclusions

Interactive visualizations for data exploration

Interactive visualizations are useful for exploring data to find trends

- They can be shared on the internet
- They can't be put in static pdfs
  - But can still be useful for your final project to find trends that you can display with static graphics

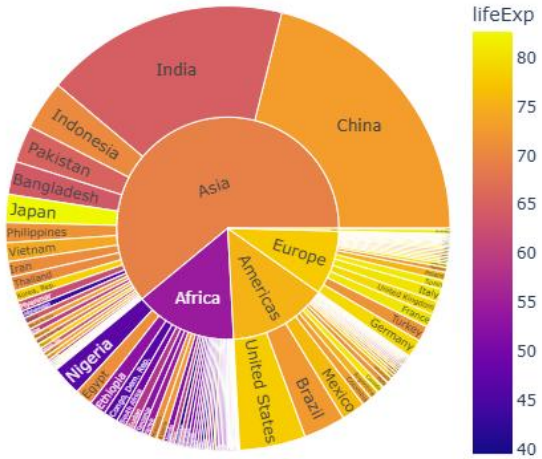
We used plotly to create interactive graphics

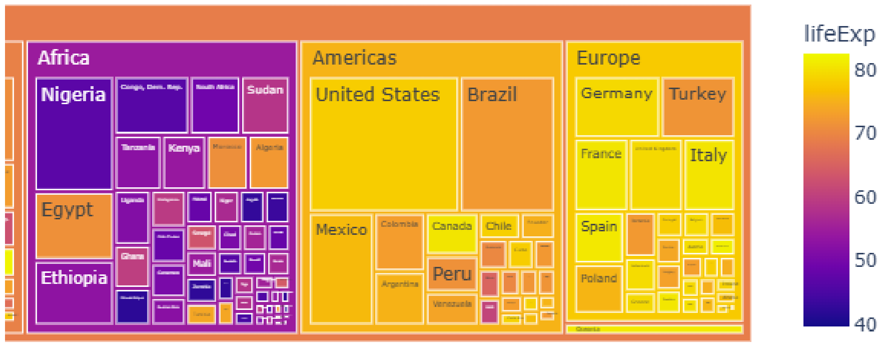
```
import plotly.express as px
```

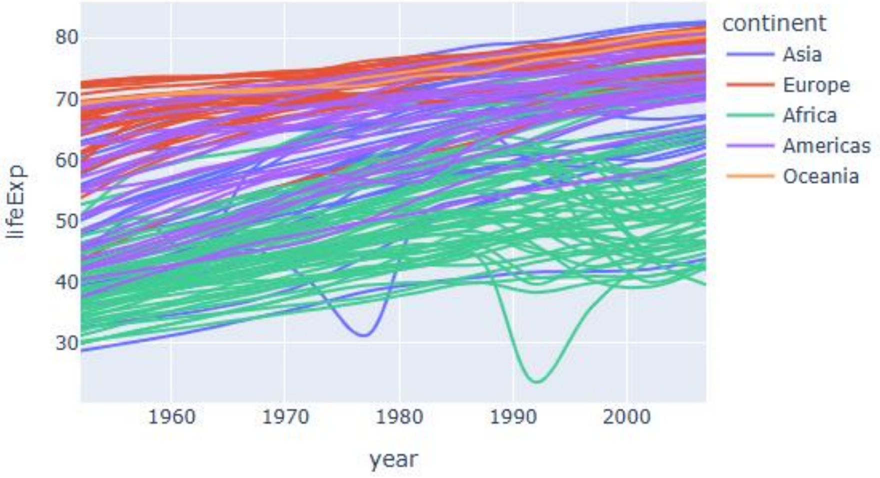
Plotly interactive plots

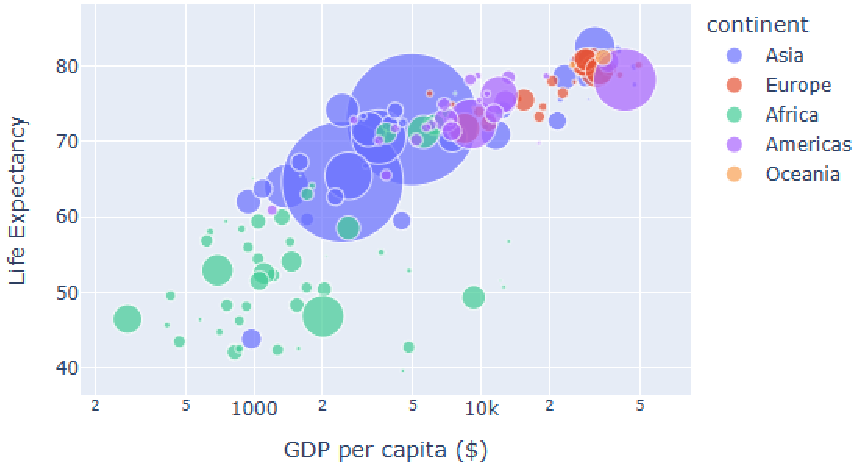
Interactive plots:

- px.line()
- px.scatter()
- px.sunburst()
- px.treemap()









Pivot Tables and heatmaps

Pivot tables aggregate values based on to two grouping variables, and create a table where:

- The rows are the levels of one cat. variable
- The columns are the levels of the second cat. variable
- The values are aggregated over a third quant. variable

```
df2 = df.pivot_table(index = "col1", columns = "col2",
                     values = "col3", aggfunc = "max")
```

```
nba2 = nba.pivot_table(index = "TEAM", columns = "POSITION",
                       values = "SALARY",  aggfunc = "max")
```

rows columns values

	PLAYER	TEAM	POSITION	SALARY
0	De'Andre Hunter	Atlanta Hawks	SF	9.835881
1	Jalen Johnson	Atlanta Hawks	SF	2.792640
2	AJ Griffin	Atlanta Hawks	SF	3.536160
3	Trent Forrest	Atlanta Hawks	SG	0.508891
4	John Collins	Atlanta Hawks	PF	23.500000

POSITION	C	PF	PG	SF	SG
TEAM					
Atlanta Hawks	18.206896	23.500000	37.096500	9.835881	17.071120
Boston Celtics	26.500000	4.306281	22.600000	30.351780	17.142857
Brooklyn Nets	9.391069	44.119845	38.917057	20.100000	19.500000
Charlotte Bobcats	3.722040	1.563518	8.623920	30.075000	21.486316
Chicago Bulls	3.200000	7.775400	9.030000	27.300000	37.096500

Pivot Tables and heatmaps

rows columns values

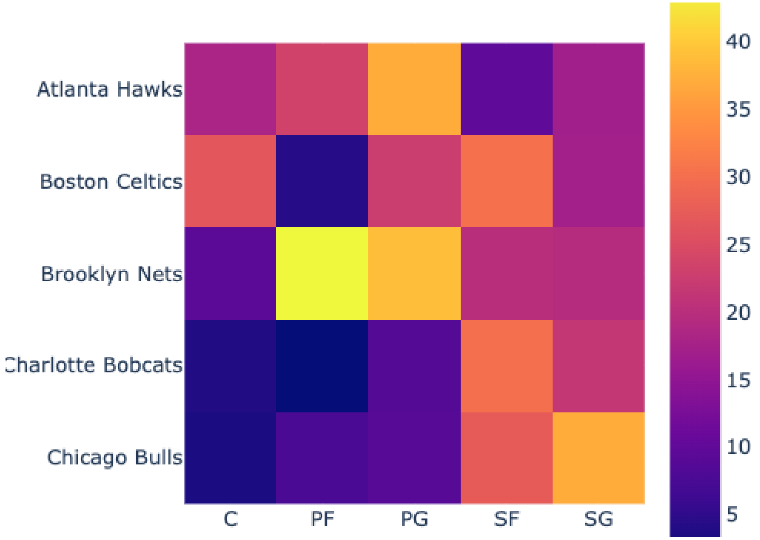
	PLAYER	TEAM	POSITION	SALARY
0	De'Andre Hunter	Atlanta Hawks	SF	9.835881
1	Jalen Johnson	Atlanta Hawks	SF	2.792640
2	AJ Griffin	Atlanta Hawks	SF	3.536160
3	Trent Forrest	Atlanta Hawks	SG	0.508891
4	John Collins	Atlanta Hawks	PF	23.500000

One can then visualize the data as a heatmap using plotly or seaborn

```
sns.heatmap(nba2)    # seaborn

px.imshow(nba2)      # plotly
```

POSITION	C	PF	PG	SF	SG
TEAM					
Atlanta Hawks	18.206896	23.500000	37.096500	9.835881	17.071120
Boston Celtics	26.500000	4.306281	22.600000	30.351780	17.142857
Brooklyn Nets	9.391069	44.119845	38.917057	20.100000	19.500000
Charlotte Bobcats	3.722040	1.563518	8.623920	30.075000	21.486316
Chicago Bulls	3.200000	7.775400	9.030000	27.300000	37.096500

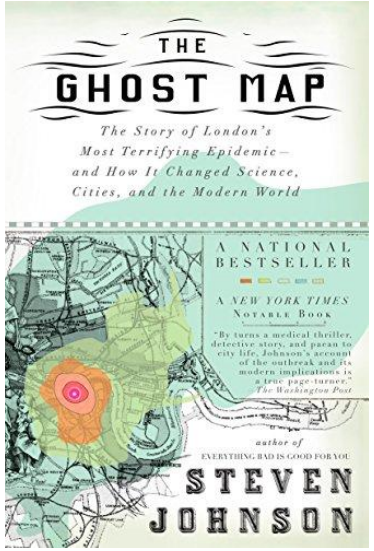


Let's explore this in Jupyter!

Maps

Maps to determine the causes of cholera

Visualizing data on a map can be a powerful way to see spatial trends



One of the first maps used to show spatial trends was created by John Snow to further his case that cholera was a water born illness

Cholera reached London in early 1830s

It was greatly feared as it was often deadly

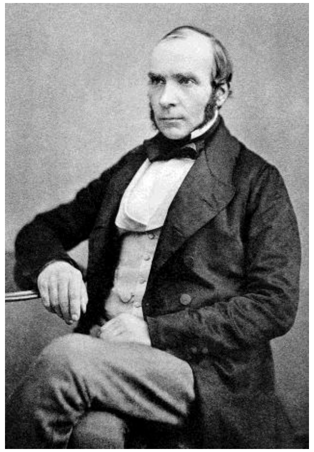
- An outbreak in 1849 killed over 14,000 people in London

Cholera in London in the 19th century

Cause of cholera was unknown. Several theories:

1. Miasmas theory: caused by bad air/smells
   - Florence Nightingale, Edwin Chadwick (board of health)

2. Water born disease
   - John Snow (anesthesiologist)

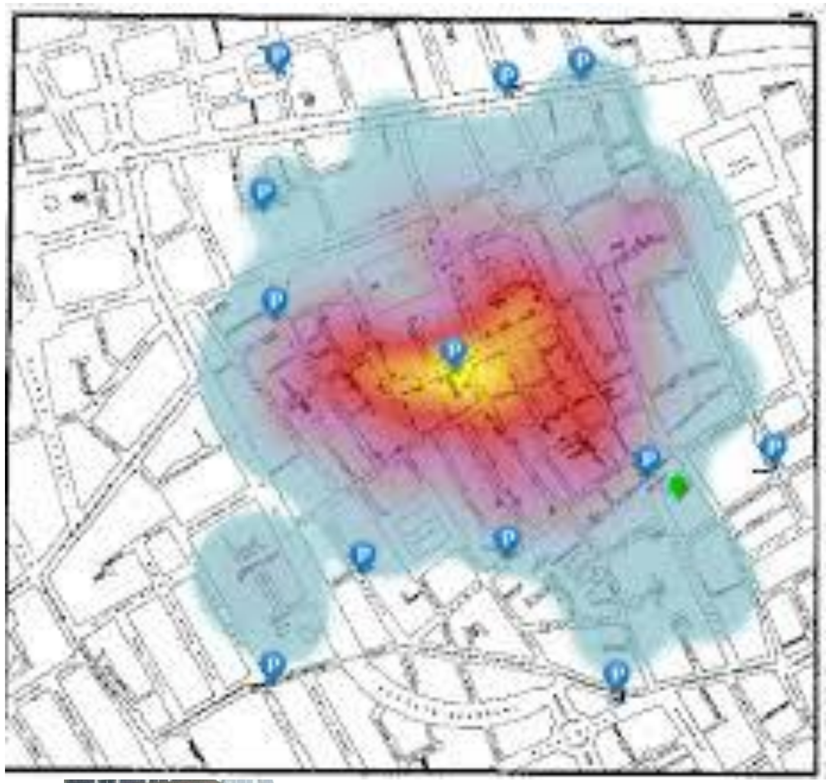


John Snow and spatial mapping

To try to understand the cause of the cholera outbreak of 1854, John Snow plotted a map of cholera deaths

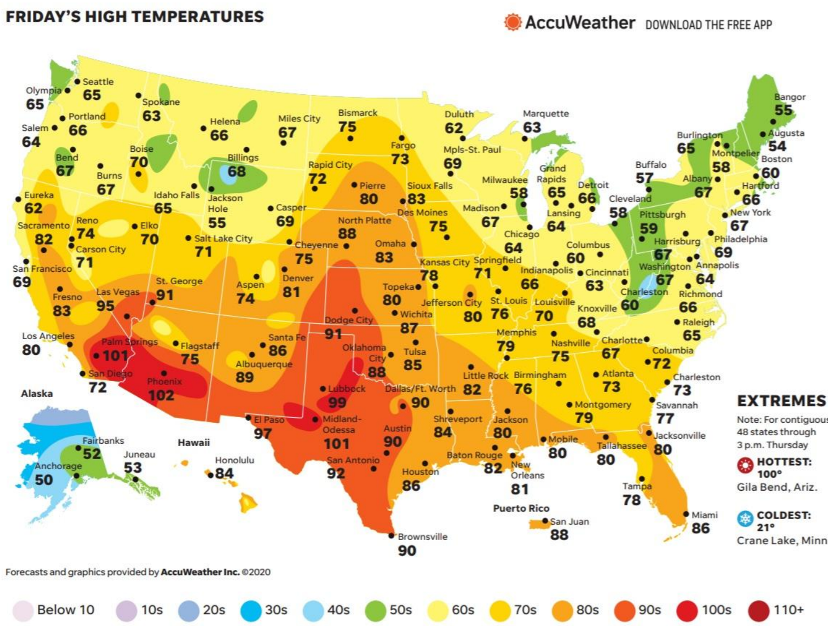
Based on this map and interviews, he concluded that the source of cholera was the Broad Steet well

- He famously removed the handle of the well to prevent the spread of disease
- Now he is considered the founder of epidemiology

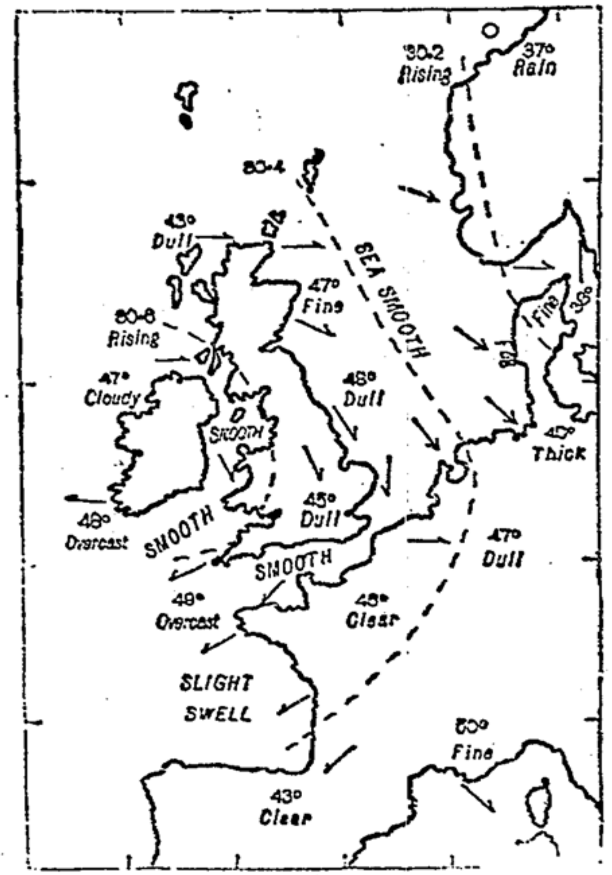


Maps

Another early use where a map gave insight was the mapping of weather by John Galton in 1875



WEATHER CHART, MARCH 31, 1875.



The dotted lines indicate the gradations of barometric pressure. The variations of the temperature are marked by figures, the state of the sea and sky by descriptive words, and the direction of the wind by arrows—barbed and feathered according to its force. ⊙ denotes calm.

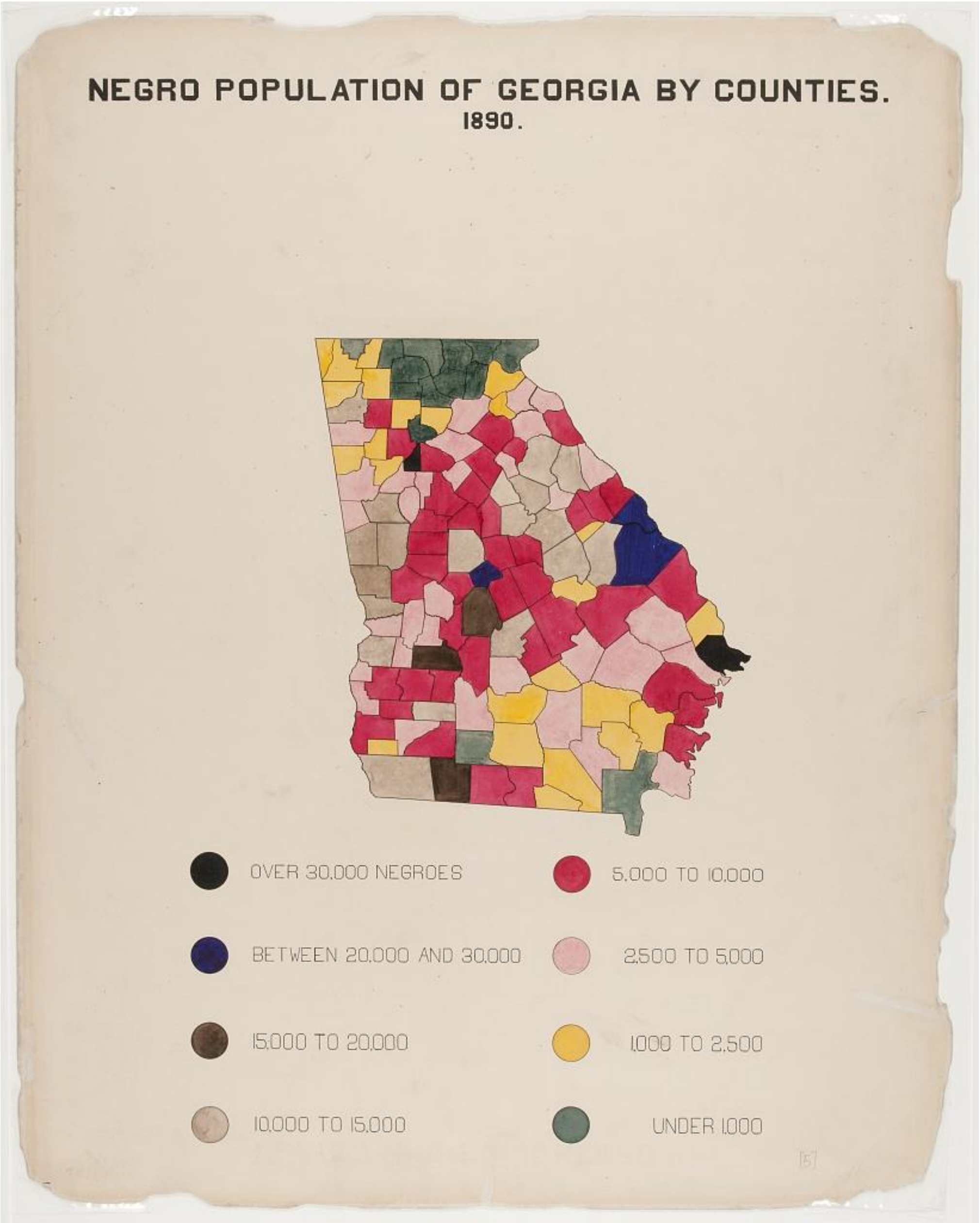
Galton's first weather map (1875)

W. E. B. Du Bois

W.E.B. Du Bois was a social scientist and prominent African-American rights activist

He gathered and manually visualized the lives of Black Americans in the 1890s

Presented 58 visualizations in the 1900 World's Fair in Paris



geopandas

To create maps in Python we will use the geopandas package

```
import geopandas as gpd
```

The key object of interest is the geopandas DataFrame

- It is the same as a regular data frame but it has an extra column called "geometry" that contains geospatial shape features
- The geometry column contrains "Shapely" objects used to represent geometric shapes

	key_comb_drvr	geometry
0	M11551	POINT (117.525391 34.008926)
1	M17307	POINT (86.51248 30.474344)
2	M19584	POINT (89.537415 37.157627)
3	M21761	POINT (117.526871 34.00647)
4	M22374	POINT (117.525345 34.008915)
5	U01997A	POINT (84.80533 33.719654)
6	U153601	POINT (78.24838 39.986454)
7	U159393	POINT (98.49438499999999 40.801544)
8	U722222	POINT (84.23309 33.9386)
9	U723030	POINT (83.86456 34.08479)
10	U723333	POINT (85.67151 42.83093)
11	U753333	POINT (117.498535 34.069157)
12	U760505	POINT (90.61252 41.456993)

geopandas

We can read in data as a geopandas DataFrame using

```
map = gpd.read_file('my_file.geojson')
```

We can plot maps using the `gpd.plot()` function

Let's explore this in Jupyter!

Coordinate reference systems

A coordinate reference system (CRS) is a framework used to precisely measure locations on the surface of the Earth as coordinates

The goal of any coordinate reference system is to create a common reference frame in which locations can be measured precisely as coordinates, so that any recipient can identify the same location that was originally intended

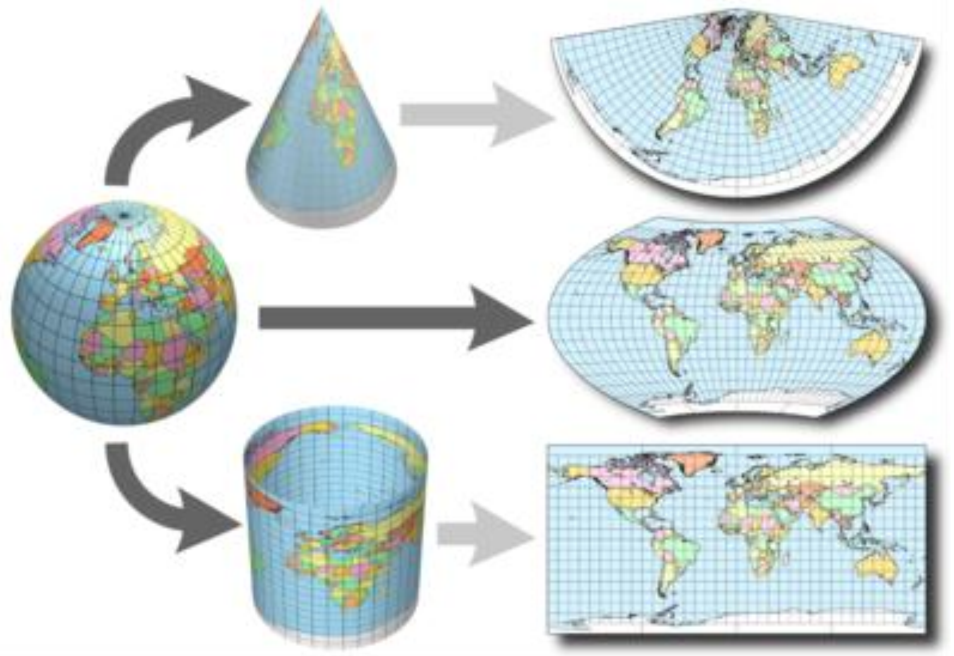
- Needed for aligning different layers on maps

Map projections

Since the earth is a 3D structure, coordinate systems have to project their data onto a 2D maps

Different projects preserve different properties

- **Mercator projection** keeps angles intact
  - Useful for navigation
- **Eckert IV projection** keeps the size of land areas intact



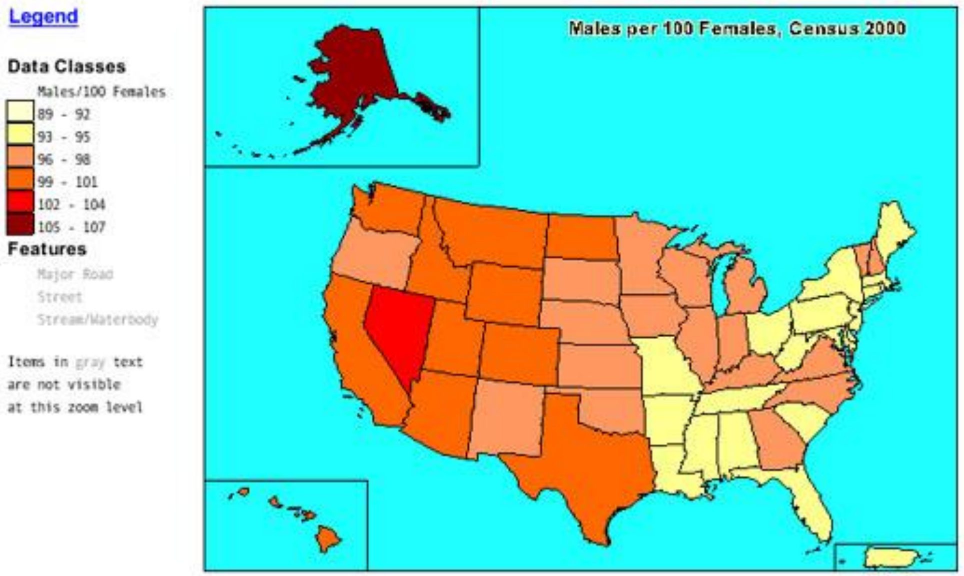
Let's explore this in Jupyter!

Maps

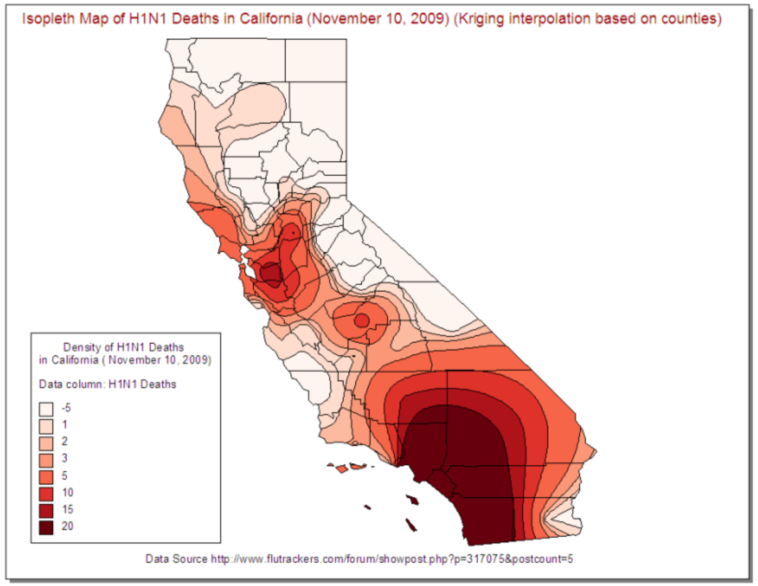
Choropleth maps: shades/colors in predefined areas based on properties of a variable

Isopleth maps: creates regions based on constant values

Choropleth map



Isopleth map



Choropleth maps

We can create choropleth maps using geopandas by joining region information on to a geopandas DataFrame that has a map

We can then use the `gpd.plot(column = )` method to visualize the map

Let's explore this in Jupyter!

Loops

For loops

For loops repeat a process many times, iterating over a sequence of items

- Often we are iterating over an array of sequential numbers

```
animals =  ["cat", "dog", "bat”]
for creature in animals:
    print(creature)

for i in np.arange(4):
    print(i**2)
```

Ranges

A range gives us a sequence of consecutive numbers

An sequence of increasing integers from 0 up to *end* - 1

- range(end)

An sequence of increasing integers from *start* up to *end* - 1

- range(start, end)

A sequence with step between consecutive values

- range(start, end, step)

The range always includes start but excludes end

Let's explore this in Jupyter!

Enumerate and zip

We can use the enumerate() function to both items in a list, and sequential integers:

```
animals =  ["cat", "dog", "bat”]
        for i, creature in enumerate(animals):
                print(i, creature)
```

We can use the zip() function to get items for two lists:

```
        animal_order =  ["feline", "canine", "chiropteran"]
        for curr_order, curr_animal in zip(animal_order, animals):
                print(curr_order, curr_animal)
```

Let's explore this in Jupyter!