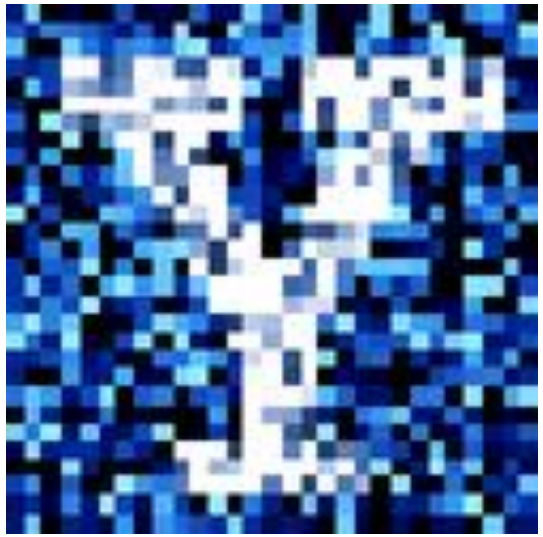
YData: Introduction to Data Science



Class 14: Interactive data visualizations

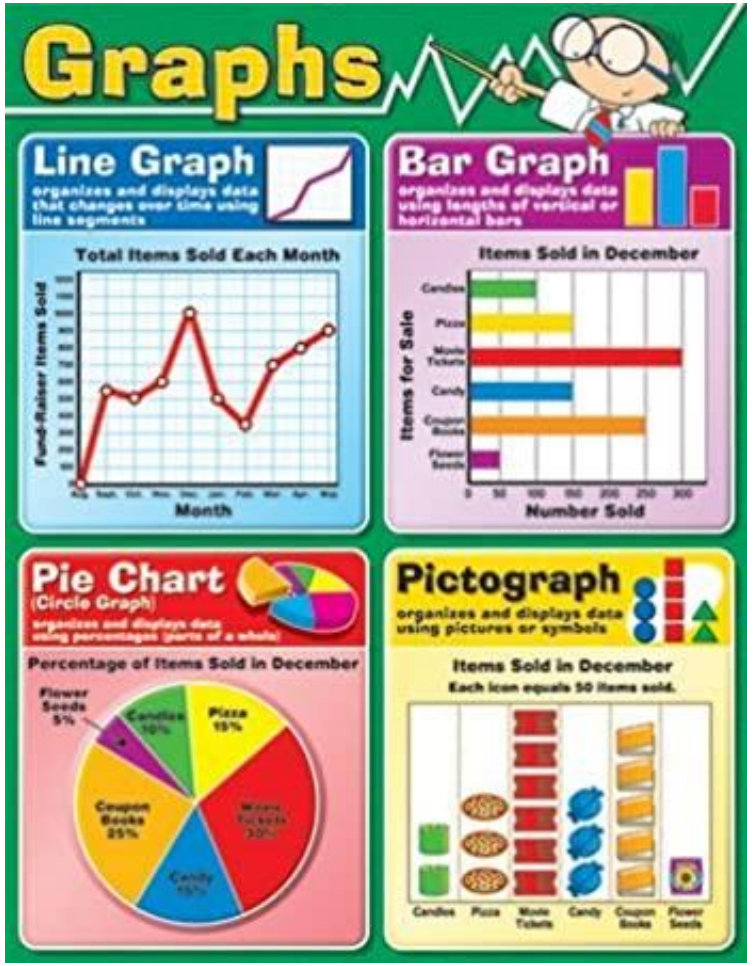
Overview

Show and tell of your visualizations from homework 5

Very quick review seaborn

Interactive graphics with plotly

If there is time: maps!



Announcement: class project

The final project is a 6-10 page Jupyter notebook report where you analyze your own data to address a question that you find interesting

- It's a chance to practice everything you've learned in class!

The goal of is to present a clear and compelling data analysis showing a few interesting results!

- Ideally something you are proud of that you could show off to potential future employers, etc.

A few sources for data sets are listed on Canvas

- You can use data you collect as well. If you use data from another class, your work must be unique for each class

Announcement: class project

A draft of the project is due on November 10th

- I'm telling you about this early so:
  - If you want to think/work on the project over break you can
  - If don't want to think about it over break you don't have to

There will be a "peer review" period where you will give and receive feedback on three other projects

- Instructions for how to do the review will be given

The final version will be due on December 8th

A Jupyter notebook "template" project will be available soon

Review of data visualization!

*Statistical projections which **speak to the senses without fatiguing the mind**, possess the advantage of fixing the attention on a great number of important facts.*

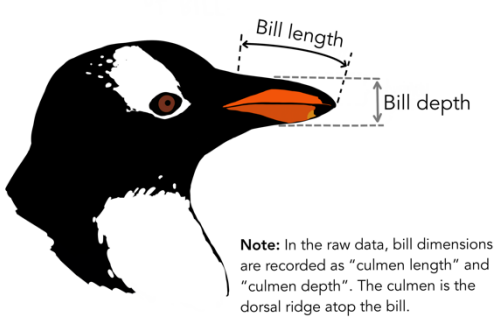
—Alexander von Humboldt, 1811

Let’s take a few minutes to explore the data visualizations you found and that you created as part of homework 5!

Quick review: Seaborn

"Seaborn is a Python data visualization library based on **matplotlib**. It provides a high-level interface for drawing attractive and informative statistical graphics."

To explore seaborn, we looked at data on penguins!




```
import seaborn as sns

sns.set_theme()
```

Seaborn figure level plots

Figure level plots are grouped based on the types of variables being plotted

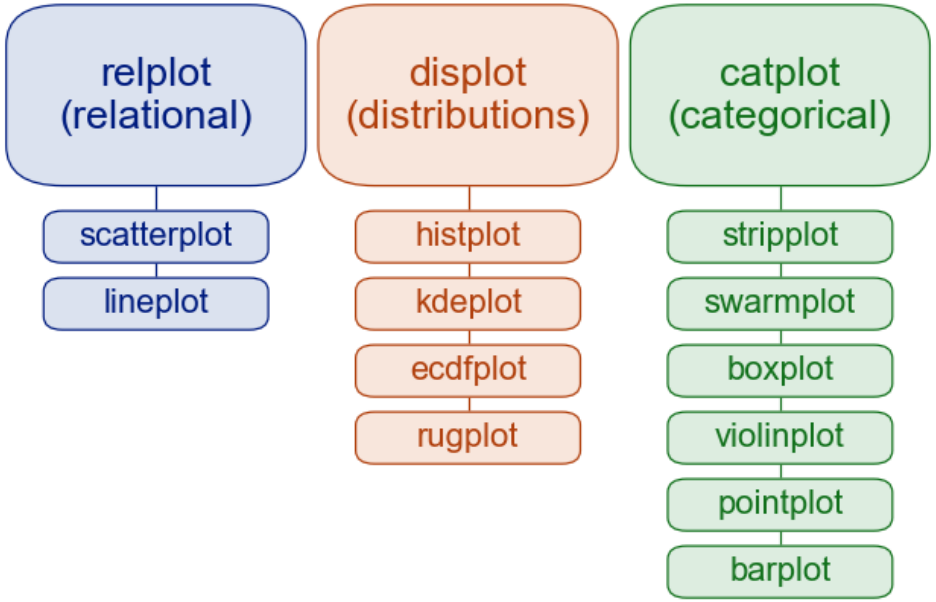
In particular, there are plots for:

1. Two quantitative variables
   - sns.relplot()

2. A single quantitative variable
   - sns.displot()

3. Quantitative variable compared across different categorical levels
   - sns.catplot()

Figure level plots



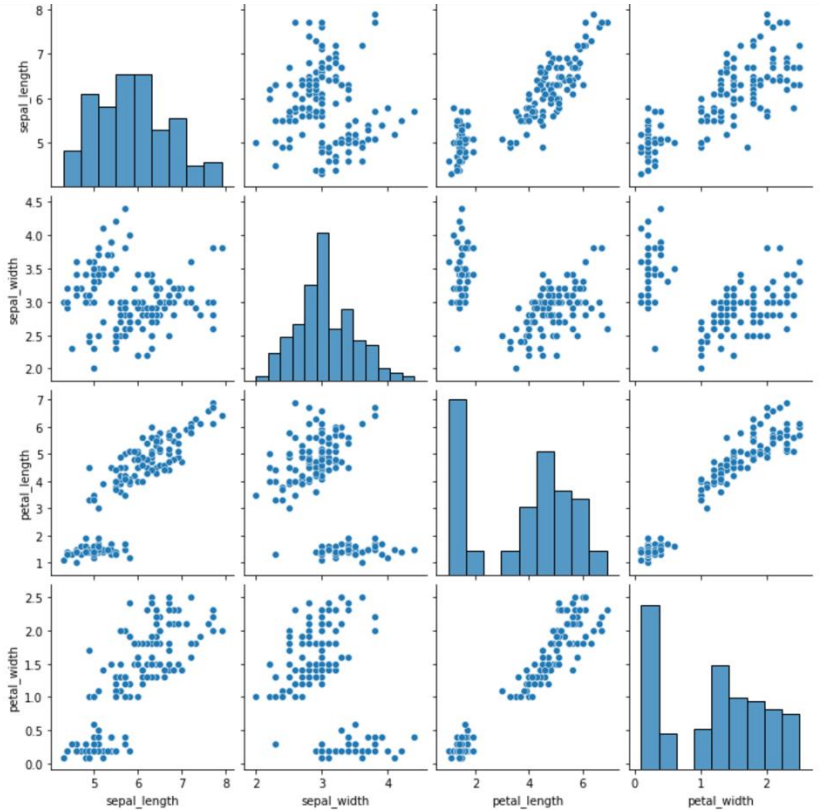
Pairs plot

A pairs plot, create scatter plots between all quantitative variables in a DataFrame

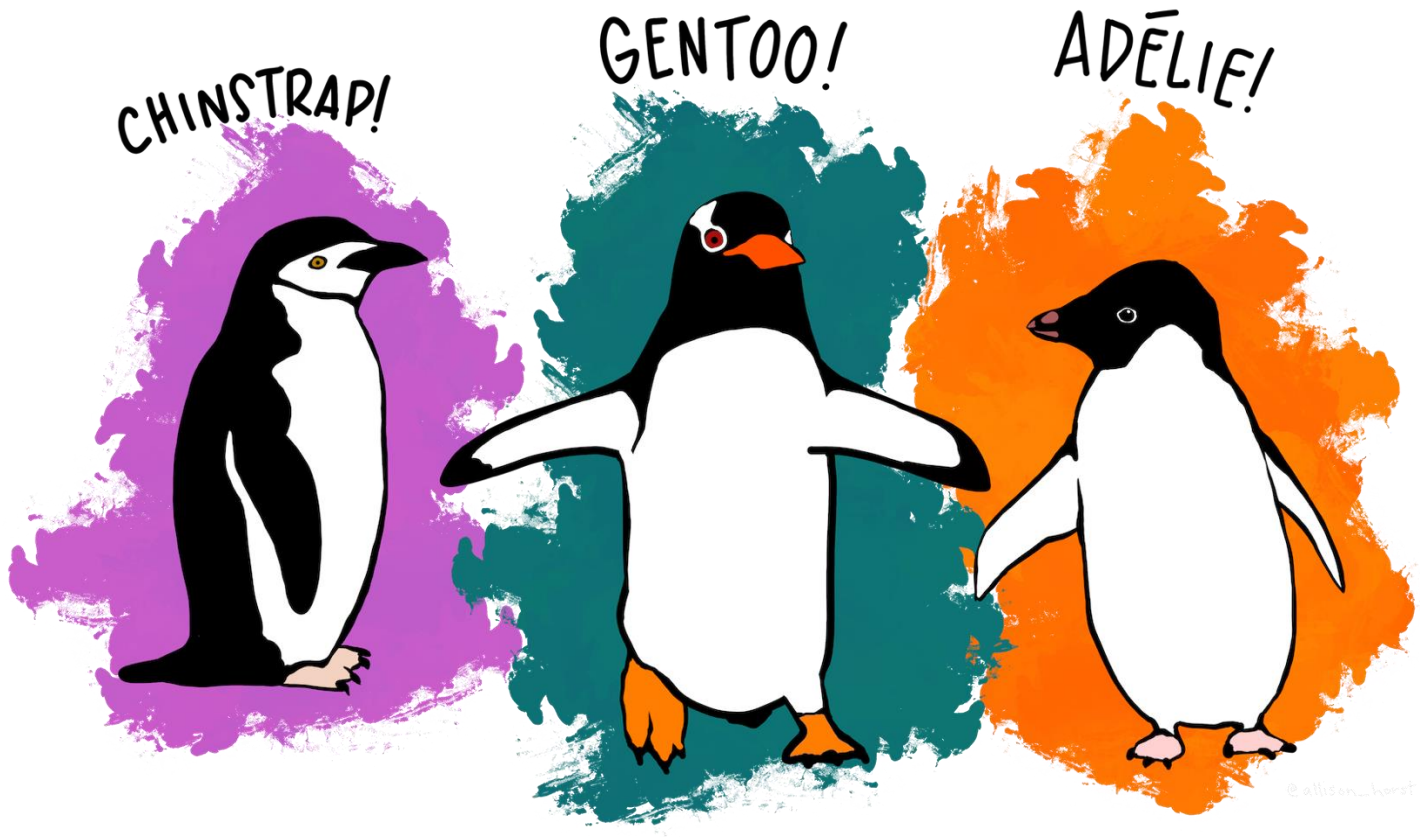
It can be one of the most useful ways to see relationships between multiple quantitative variabels

We can create pairs plots in seaborn using:

```
sns.pairplot(the_data)
```



Let’s do a quick warm-up exercise!


CHINSTRAP!
GENTOO!
ADÉLIE!

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 will use plotly to create interactive graphics

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

Plotly interactive plots

Line plots

```
fig = px.line(data_frame =, x =, y = , color = , hover_name = , line_shape = )
```

Scatter plots

```
fig = px.scatter(data_frame = , x = , y = , size = , color = , hover_name = )
```

Add axis labels

```
fig.update_layout(xaxis_title="X", yaxis_title="Y")
```

Let's explore this in Jupyter!

Plotly interactive plots

Sunburst plots

```
px.sunburst(data_frame = ,  path = , values = ,  color = )
```

Treemap

```
px.treemap(data_frame = , path = , values = , color =  )
```

Let's explore this in Jupyter!

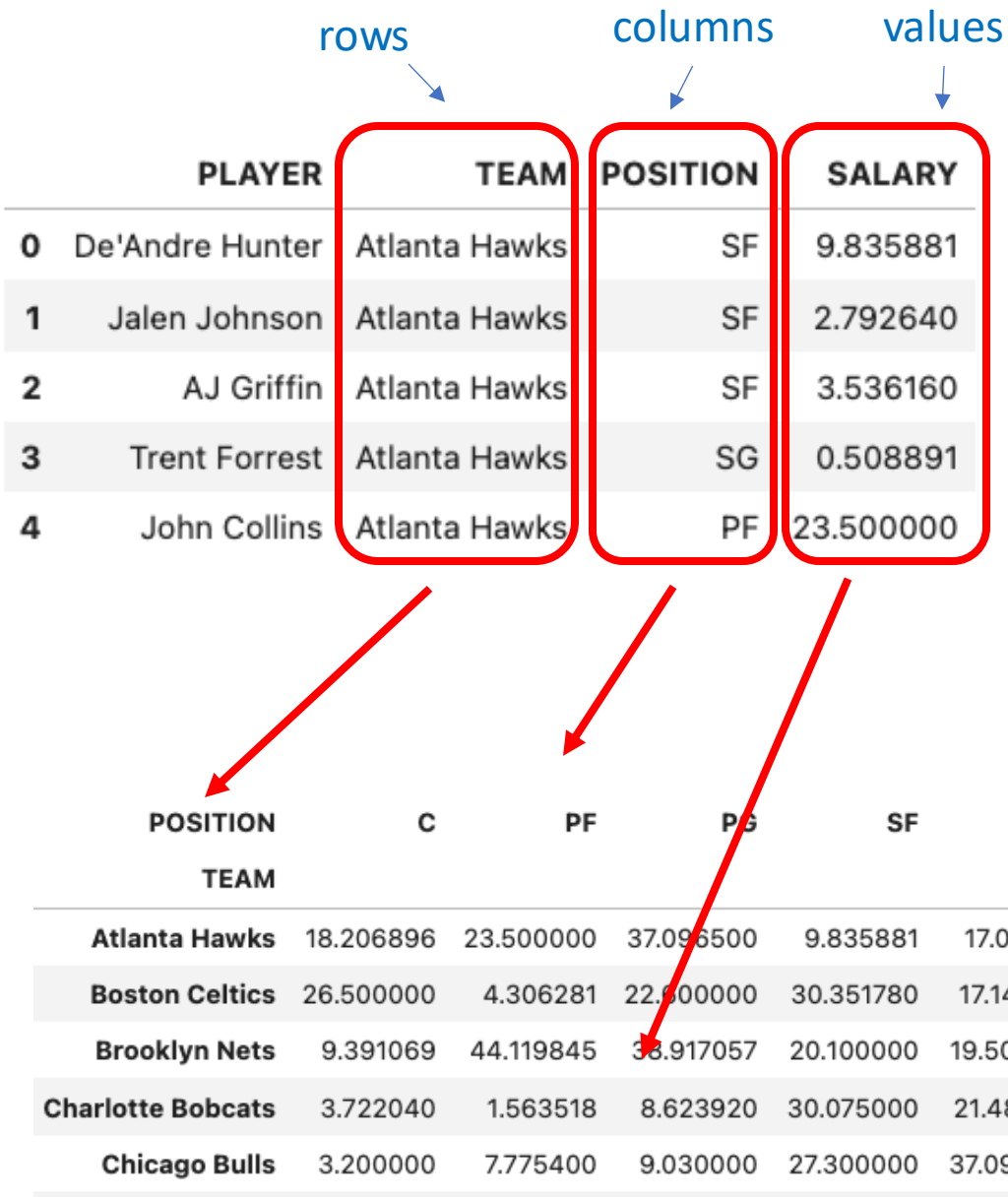
Pivot Tables and heatmaps

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”)
```



	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

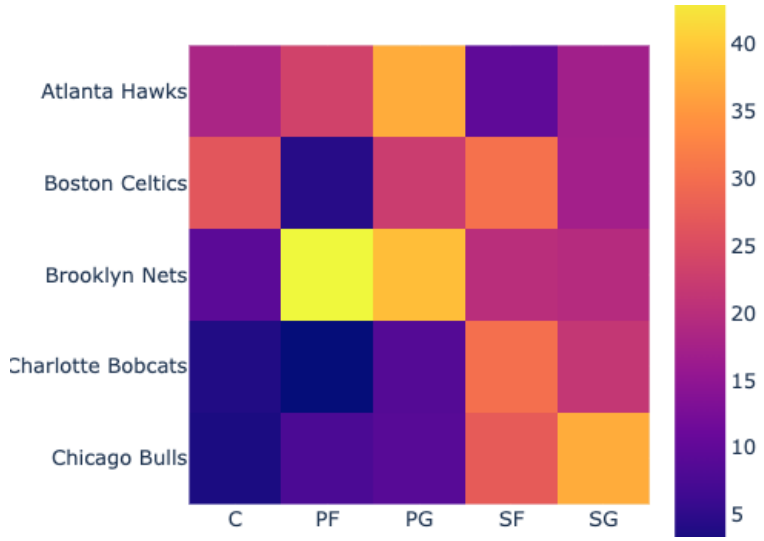
POSITION TEAM	C	PF	PG	SF	SG
Atlanta Hawks	18.206896	23.500000	37.096500	9.835881	17.071120
Boston Celtics	26.500000	4.306281	22.500000	30.351780	17.142857
Brooklyn Nets	9.391069	44.119845	38.917057	20.100000	19.500000
Charlotte Bobcats	3.722040	1.563518	8.623920	30.075000	21.486316
Chicago Bulls	3.200000	7.775400	9.030000	27.300000	37.096500

Pivot Tables and heatmaps

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

px.imshow(nba2) # plotly

sns.heatmap(nba2) # seaborn



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 TEAM	C	PF	PG	SF	SG
Atlanta Hawks	18.206896	23.500000	37.096500	9.835881	17.071120
Boston Celtics	26.500000	4.306281	22.500000	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

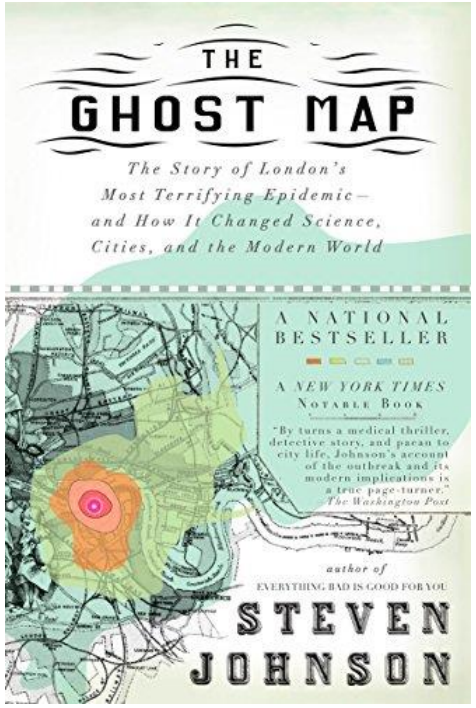
Let's explore this in Jupyter!

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 in London in the 19th century

Cholera reached London in early 1830s

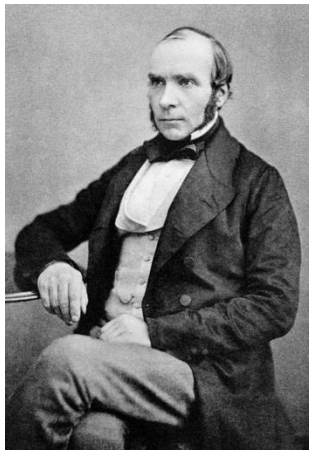
It was greatly feared as it was often deadly

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

Cause of cholera was unknown. Several theories:

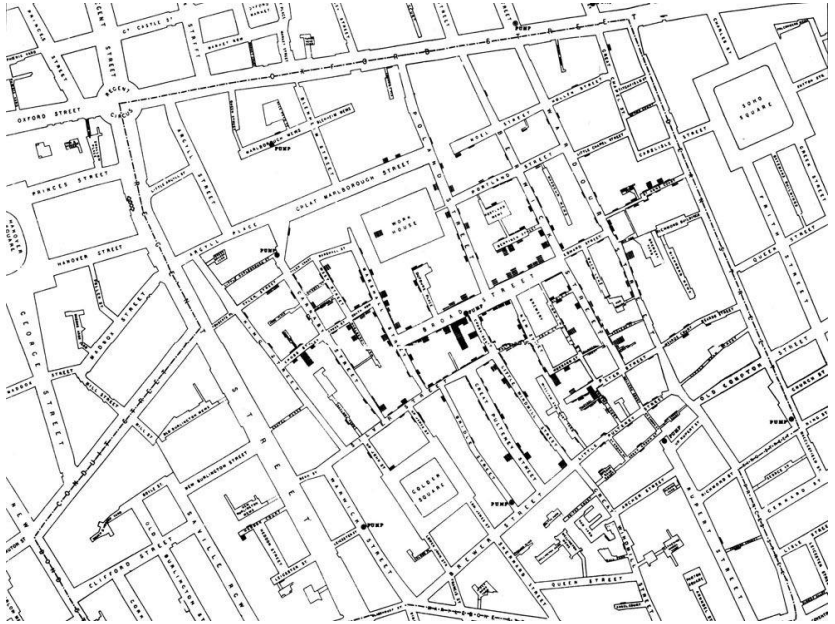
1. Miasmas theory: caused by bad air/smells
   - Florence Nightingale, Edwin Chadwick (board of health)

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

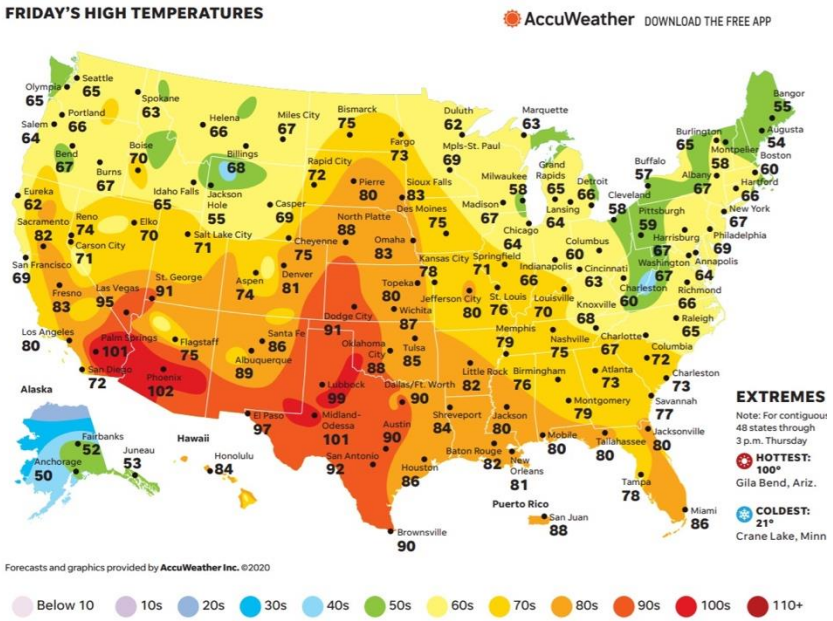


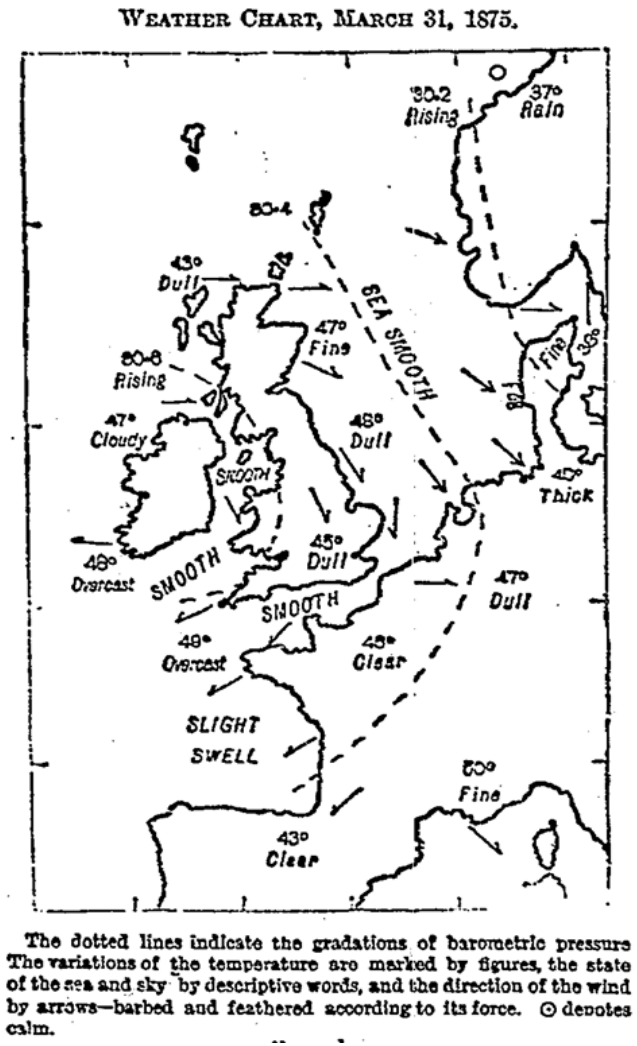
Based on this map and interviews, he concluded that the source of cholera was the Broad Steet well

- 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





Galton's first weather map (1875)

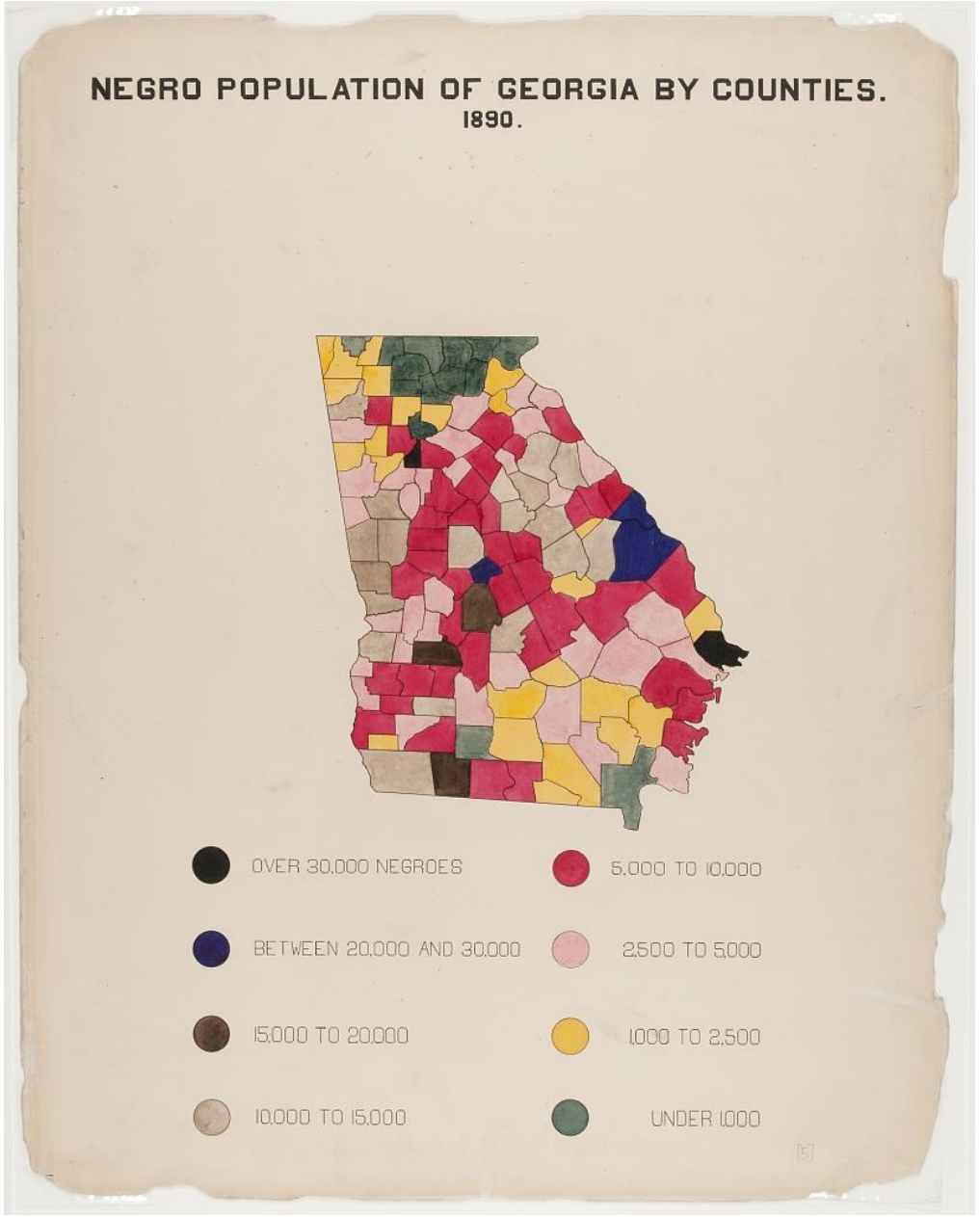
W. E. B. Du Bois

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

Took on the complex task of gathering and manually visualizing the lives of Black Americans in the 1890s

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

- Won a gold award



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

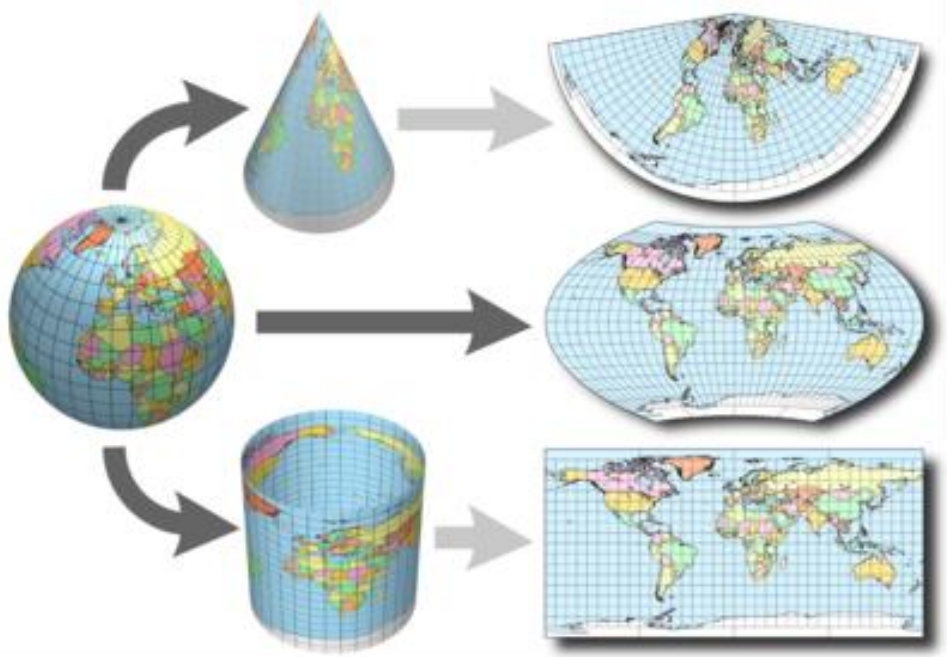
A coordinate reference system (CRS) is a framework used to precisely measure locations on the surface of the Earth as coordinates

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!

WHAT YOUR FAVORITE MAP PROJECTION SAYS ABOUT YOU

MERCATOR

YOU'RE NOT REALLY INTO MAPS.

VAN DER GRINTEN

YOU'RE NOT A COMPLICATED PERSON. YOU LOVE THE MERCATOR PROJECTION; YOU JUST WISH IT WEREN'T SQUARE. THE EARTH'S NOT A SQUARE, IT'S A CIRCLE. YOU LIKE CIRCLES. TODAY IS GONNA BE A GOOD DAY!

ROBINSON

YOU HAVE A COMFORTABLE PAIR OF RUNNING SHOES THAT YOU WEAR EVERYWHERE. YOU LIKE COFFEE AND ENJOY THE BEATLES. YOU THINK THE ROBINSON IS THE BEST-LOOKING PROJECTION, HANDS DOWN.

DYMAXION

YOU LIKE ISAAC ASIMOV, XML, AND SHOES WITH TOES. YOU THINK THE SEGWAY GOT A BAD RAP. YOU OWN 3D GOGGLES, WHICH YOU USE TO VIEW ROTATING MODELS OF BETTER 3D GOGGLES. YOU TYPE IN DVORAK.

WINKEL-TRIPEL

NATIONAL GEOGRAPHIC ADOPTED THE WINKEL-TRIPEL IN 1998, BUT YOU'VE BEEN A WT FAN SINCE *LONG* BEFORE 'NATGEO' SHOWED UP. YOU'RE WORRIED IT'S GETTING PLAYED OUT, AND ARE THINKING OF SWITCHING TO THE KAVRAYSKIY. YOU ONCE LEFT A PARTY IN DISGUST WHEN A GUEST SHOWED UP WEARING SHOES WITH TOES. YOUR FAVORITE MUSICAL GENRE IS "POST-".

GOODE HOMOLOSINE

THEY SAY MAPPING THE EARTH ON A 2D SURFACE IS LIKE FLATTENING AN ORANGE PEEL, WHICH SEEMS EASY ENOUGH TO YOU. YOU LIKE EASY SOLUTIONS. YOU THINK WE WOULDN'T HAVE SO MANY PROBLEMS IF WE'D JUST ELECT *NORMAL* PEOPLE TO CONGRESS INSTEAD OF POLITICIANS. YOU THINK AIRLINES SHOULD JUST BUY FOOD FROM THE RESTAURANTS NEAR THE GATES AND SERVE *THAT* ON BOARD. YOU CHANGE YOUR CAR'S OIL, BUT SECRETLY WONDER IF YOU REALLY *NEED* TO.

HOBO-DYER

YOU WANT TO AVOID CULTURAL IMPERIALISM, BUT YOU'VE HEARD BAD THINGS ABOUT GALL-PETERS. YOU'RE CONFLICT-AVERSE AND BUY ORGANIC. YOU USE A RECENTLY-INVENTED SET OF GENDER-NEUTRAL PRONOUNS AND THINK THAT WHAT THE WORLD NEEDS IS A REVOLUTION IN CONSCIOUSNESS.

A GLOBE!

YES, YOU'RE VERY CLEVER.

PEIRCE QUINCUNCIAL

YOU THINK THAT WHEN WE LOOK AT A MAP, WHAT WE REALLY SEE IS OURSELVES. AFTER YOU FIRST SAW *INCEPTION*, YOU SAT SILENT IN THE THEATER FOR SIX HOURS. IT FREAKS YOU OUT TO REALIZE THAT EVERYONE AROUND YOU HAS A SKELETON INSIDE THEM. YOU *HAVE* REALLY LOOKED AT YOUR HANDS.

PLATE CARRÉE

(EQUIRECTANGULAR)

YOU THINK THIS ONE IS FINE. YOU LIKE HOW X AND Y MAP TO LATITUDE AND LONGITUDE. THE OTHER PROJECTIONS OVERCOMPLICATE THINGS. YOU WANT ME TO STOP ASKING ABOUT MAPS SO YOU CAN ENJOY DINNER.

WATERMAN BUTTERFLY

REALLY? YOU KNOW THE WATERMAN? HAVE YOU SEEN THE 1909 CAHILL MAP IT'S BASED—... YOU HAVE A FRAMED REPRODUCTION AT HOME?! WHOA. ...LISTEN, FORGET THESE QUESTIONS. ARE YOU DOING ANYTHING TONIGHT?

GALL-PETERS

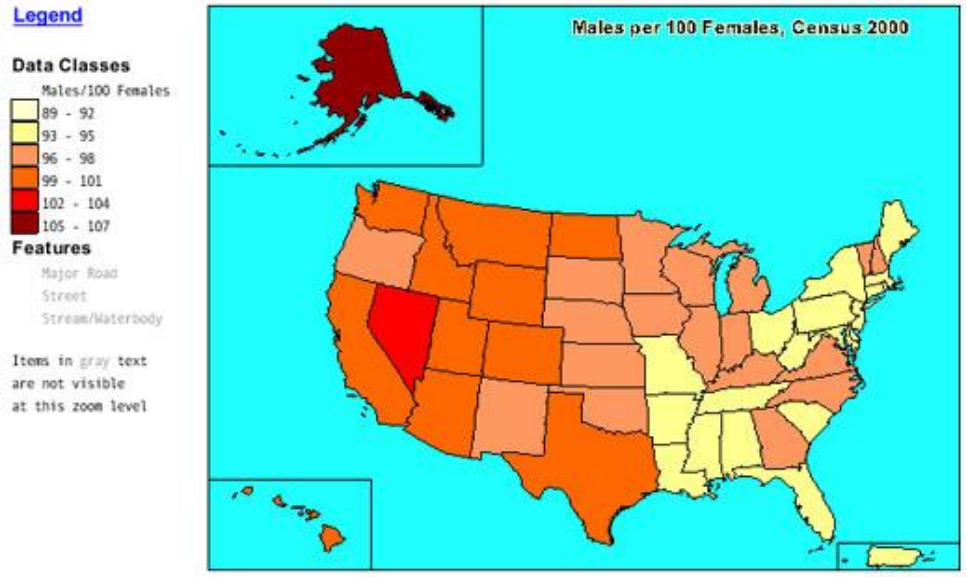
I *HATE* YOU.

Maps

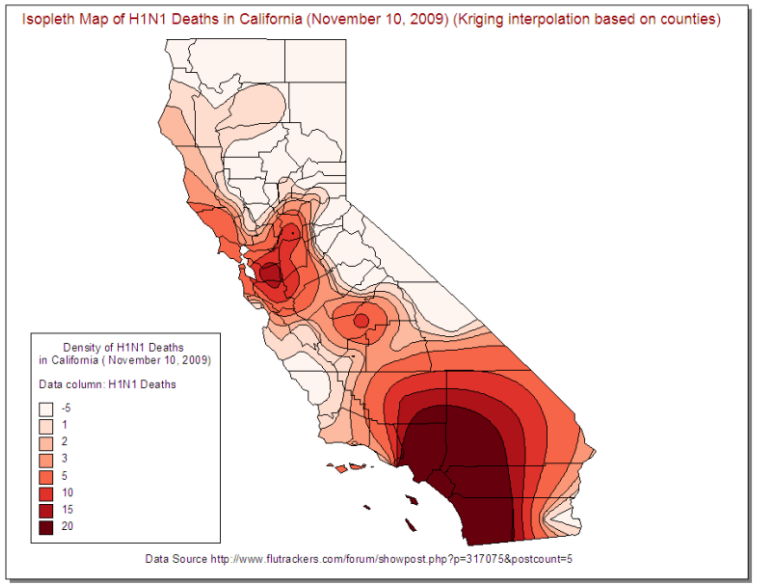
Choropleth maps: shades/colors in predefined areas based on properties of a variable

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!

Cloropleth maps can sometimes be misleading