PUBPOL 2130/ INFO 3130

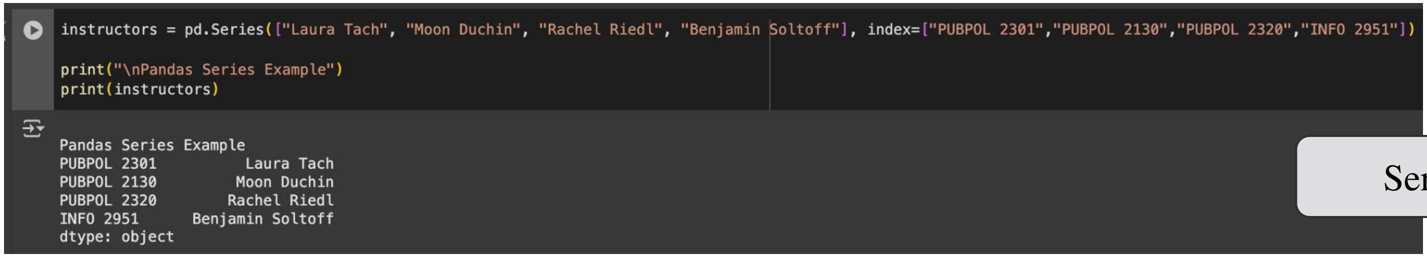
Lab 1

Logistics!

- Main GitHub Repository here
- We suggest going through the gentle intro notebook if you do not have experience with programming concepts
- You can reach out to us at
  - jrg377@cornell.edu
  - tp399@cornell.edu
- You can use Colab, Jupyter, VS code, etc.
- We're not going to do installations today, let's work on Colab if you don't have Jupyter, VS Code, etc. installed

Jan 24: Pandas

Python library used for data manipulation, analysis, and cleaning.

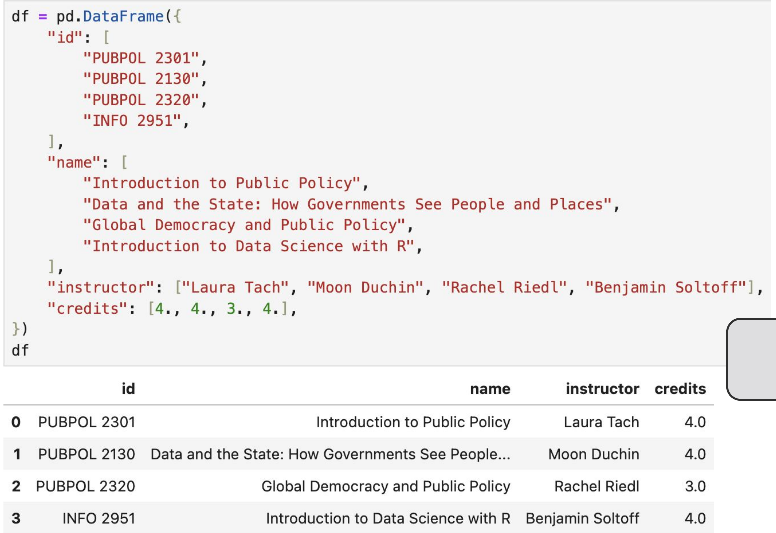


```
instructors = pd.Series(["Laura Tach", "Moon Duchin", "Rachel Riedl", "Benjamin Soltoff"], index=["PUBPOL 2301","PUBPOL 2130","PUBPOL 2320","INFO 2951"])

print("\nPandas Series Example")
print(instructors)
```

```
Pandas Series Example
PUBPOL 2301          Laura Tach
PUBPOL 2130         Moon Duchin
PUBPOL 2320        Rachel Riedl
INFO 2951      Benjamin Soltoff
dtype: object
```

Series



```
df = pd.DataFrame({
    "id": [
        "PUBPOL 2301",
        "PUBPOL 2130",
        "PUBPOL 2320",
        "INFO 2951",
    ],
    "name": [
        "Introduction to Public Policy",
        "Data and the State: How Governments See People and Places",
        "Global Democracy and Public Policy",
        "Introduction to Data Science with R",
    ],
    "instructor": ["Laura Tach", "Moon Duchin", "Rachel Riedl", "Benjamin Soltoff"],
    "credits": [4., 4., 3., 4.],
})
df
```

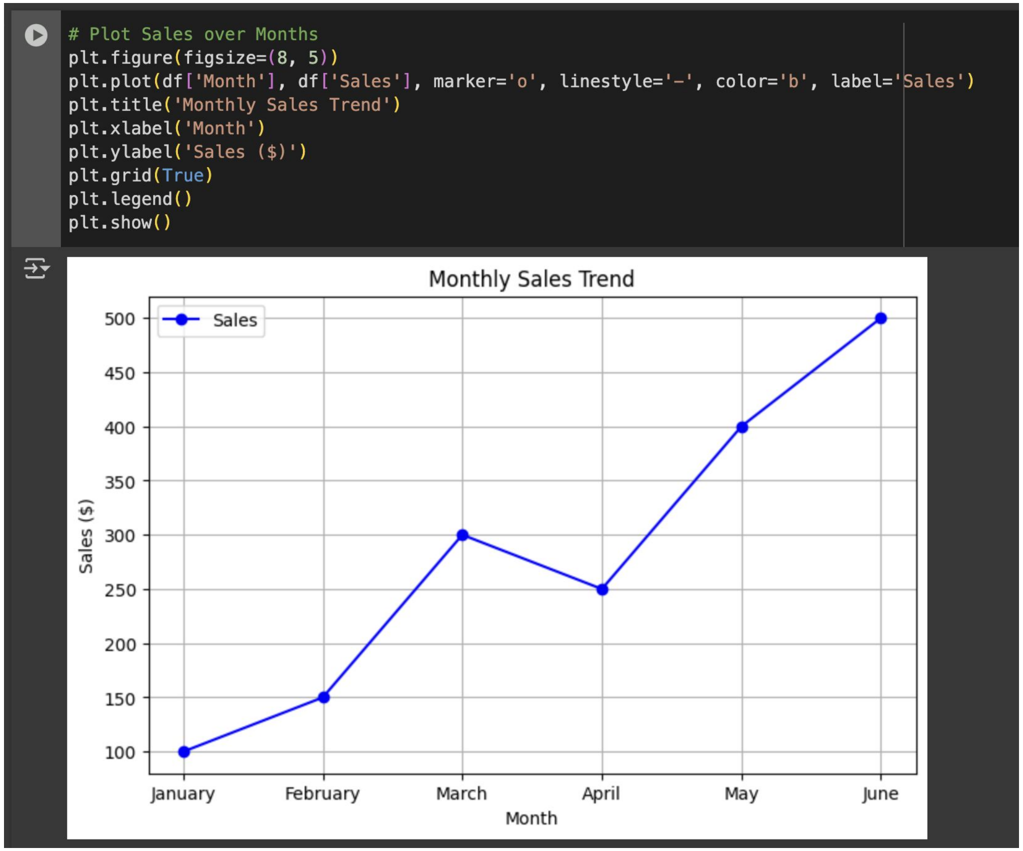
	id	name	instructor	credits
0	PUBPOL 2301	Introduction to Public Policy	Laura Tach	4.0
1	PUBPOL 2130	Data and the State: How Governments See People...	Moon Duchin	4.0
2	PUBPOL 2320	Global Democracy and Public Policy	Rachel Riedl	3.0
3	INFO 2951	Introduction to Data Science with R	Benjamin Soltoff	4.0

Dataframe

Jan 24: Matplotlib Theory

Python library used for creating static, interactive, and animated visualizations.

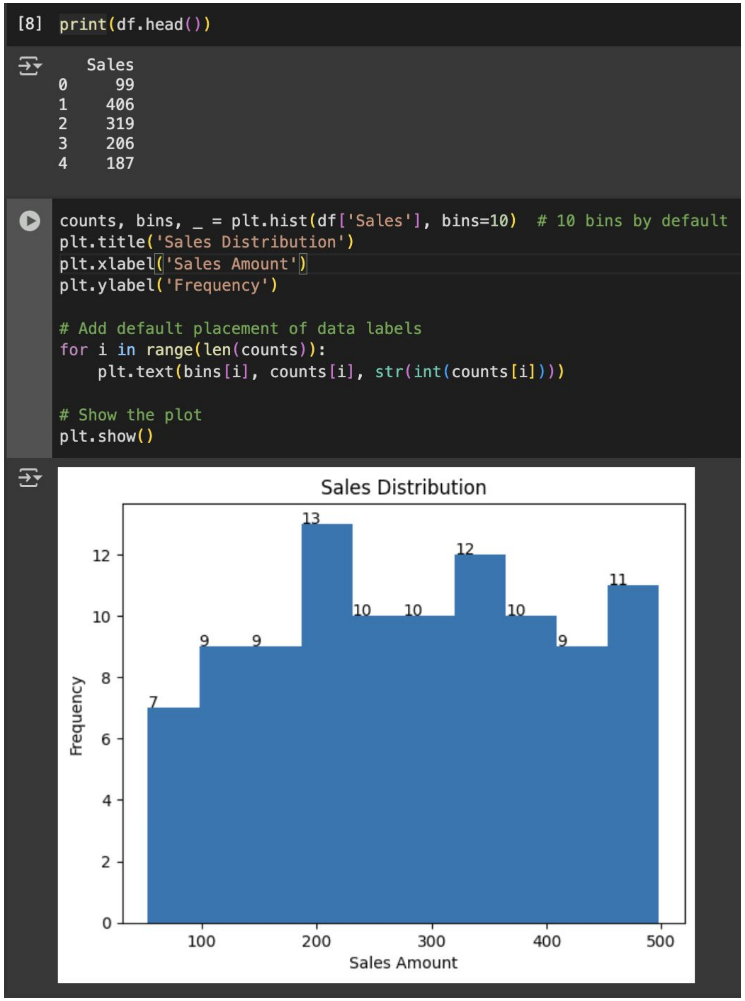
- Versatility
- Customization
- Integration
- Interactive Capabilities
- Export Options



```
# Plot Sales over Months
plt.figure(figsize=(8, 5))
plt.plot(df['Month'], df['Sales'], marker='o', linestyle='-', color='b', label='Sales')
plt.title('Monthly Sales Trend')
plt.xlabel('Month')
plt.ylabel('Sales ($)')
plt.grid(True)
plt.legend()
plt.show()
```

Jan 24: Matplotlib Disadvantages

- Complexity
- Verbose Syntax
- Limited Interactivity
- Performance Issues
- Default Aesthetics are Outdated



```
[8] print(df.head())
```

```
   Sales
0     99
1    406
2    319
3    206
4    187
```

```
counts, bins, _ = plt.hist(df['Sales'], bins=10)  # 10 bins by default
plt.title('Sales Distribution')
plt.xlabel('Sales Amount')
plt.ylabel('Frequency')

# Add default placement of data labels
for i in range(len(counts)):
    plt.text(bins[i], counts[i], str(int(counts[i])))

# Show the plot
plt.show()
```

Let’s start executing week1.ipynb together!

PUBPOL 2130/ INFO 3130

Lab 2

Announcements!

Weekly homework assignments:

- Will be due in 11 days
- New homework assigned on Fridays during lab
- Turn in on Gradescope

Upcoming exam on Feb. 13th:

- Will be 40 minutes, in class
- Lecture on Feb. 11th – likely exam review or makeup

Announcements!

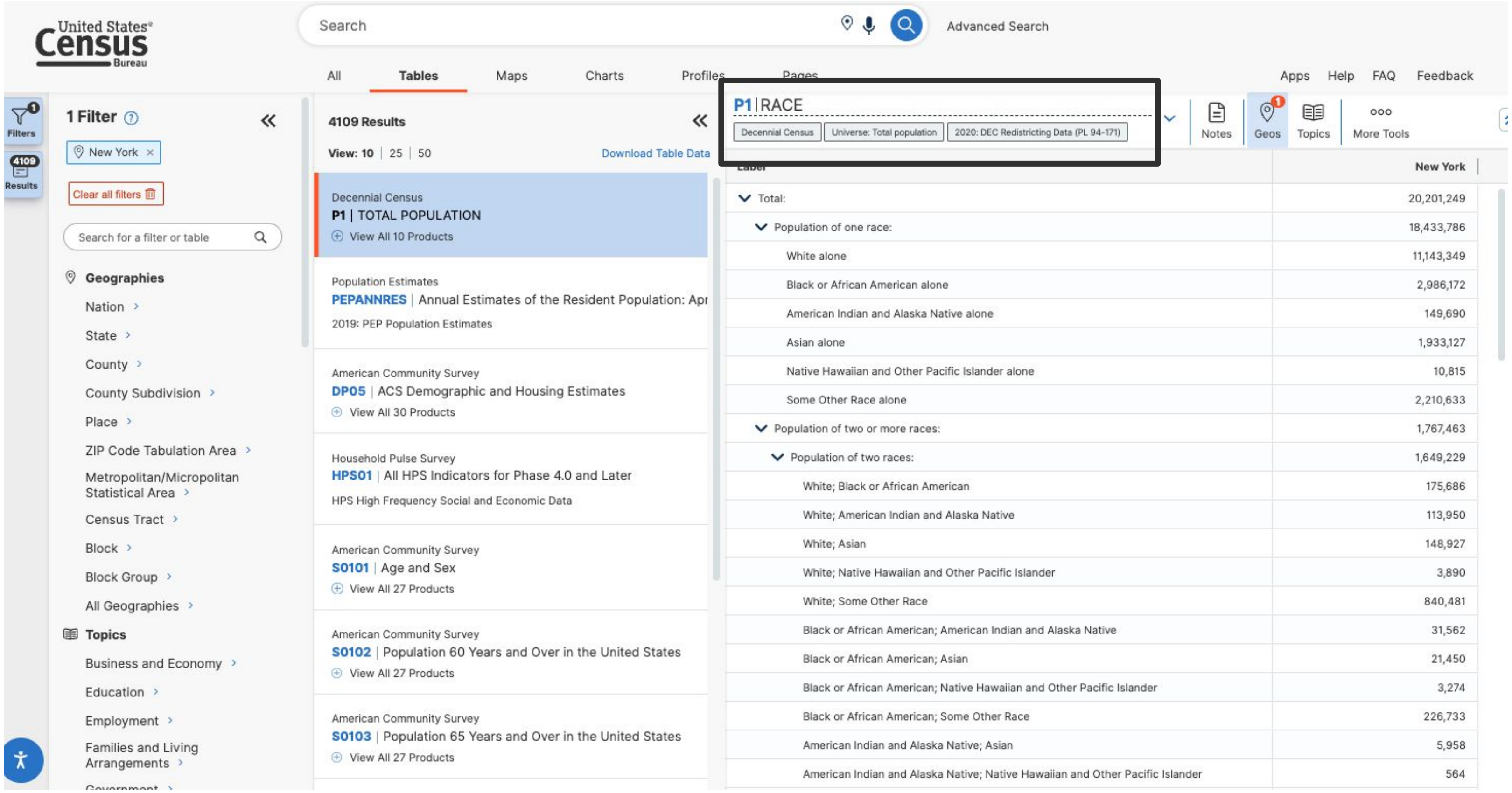
Homework Reminders:

- Don’t give us code unless we ask for it!
  - *Don’t turn in an .ipynb file*
  - *Turn in exports, not screenshots*

- Make sure axis labels are clear
- Include information on parameters that don’t change

- Default parameters in matplotlib may not be optimal – experiment with different ones
  - *E.g., binning with histograms*

Jan 31: Census Data



4109 Results

View: 10 | 25 | 50

Download Table Data

Decennial Census
P1 | TOTAL POPULATION
View All 10 Products

Population Estimates
PEPANNRES | Annual Estimates of the Resident Population: Apr
2019: PEP Population Estimates

American Community Survey
DP05 | ACS Demographic and Housing Estimates
View All 30 Products

Household Pulse Survey
HPS01 | All HPS Indicators for Phase 4.0 and Later
HPS High Frequency Social and Economic Data

American Community Survey
S0101 | Age and Sex
View All 27 Products

American Community Survey
S0102 | Population 60 Years and Over in the United States
View All 27 Products

American Community Survey
S0103 | Population 65 Years and Over in the United States
View All 27 Products

P1 | RACE

Decennial Census | Universe: Total population | 2020: DEC Redistricting Data (PL 94-171)

Label	New York
Total:	20,201,249
Population of one race:	18,433,786
White alone	11,143,349
Black or African American alone	2,986,172
American Indian and Alaska Native alone	149,690
Asian alone	1,933,127
Native Hawaiian and Other Pacific Islander alone	10,815
Some Other Race alone	2,210,633
Population of two or more races:	1,767,463
Population of two races:	1,649,229
White; Black or African American	175,686
White; American Indian and Alaska Native	113,950
White; Asian	148,927
White; Native Hawaiian and Other Pacific Islander	3,890
White; Some Other Race	840,481
Black or African American; American Indian and Alaska Native	31,562
Black or African American; Asian	21,450
Black or African American; Native Hawaiian and Other Pacific Islander	3,274
Black or African American; Some Other Race	226,733
American Indian and Alaska Native; Asian	5,958
American Indian and Alaska Native; Native Hawaiian and Other Pacific Islander	564

Jan 31: `census` Python Package

- Wrapper for the United States Census Bureau’s API
  - More information here
- Information on the Census Bureau API is here and here
  - You can request an API key here

```
from census import Census
from us import states

c = Census("MY_API_KEY")
c.acs5.get(('NAME', 'B25034_010E'),
           {'for': 'state:{}'.format(states.MD.fips)})
```

Note: you do not need an API key for querying small quantities of data, with minimal restrictions (e.g. <500 queries/day per IP)

Jan 31: Exporting plots

- Tricky in Colab vs. VSCode/Jupyter
- In **matplotlib**: `plt.savefig("file_name.jpg")`
- In **Colab**:

```
from google.colab import files
plt.savefig("file_name.jpg")
files.download("file_name.jpg")
```

  - Alternatively, you can use simple scripts in Colab to save exports to your **temporary** Colab environment

```
plt.savefig("file_name.jpg", format="jpeg", dpi=95)
```

Let’s start executing Week2.ipynb together!

PUBPOL 2130/ INFO 3130

Lab 3

Feb 07: What Are Shapefiles?

A shapefile is a widely used **geospatial data format** for mapping locations, boundaries, and spatial relationships.

- It represents **geographic features** as points, lines, or polygons.
- **Common Uses:** Political boundaries, census tracts, roads, environmental features.
- Shapefile **Components**:
  - .shp – Stores geometry (the actual shapes).
  - .shx – Index for quick lookup.
  - .dbf – Attribute data (tabular information).

Feb 07: Census Shapefiles

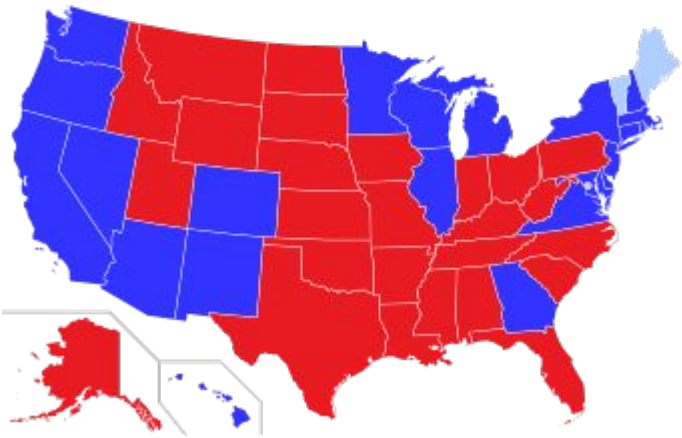
Some examples of Census shapefiles

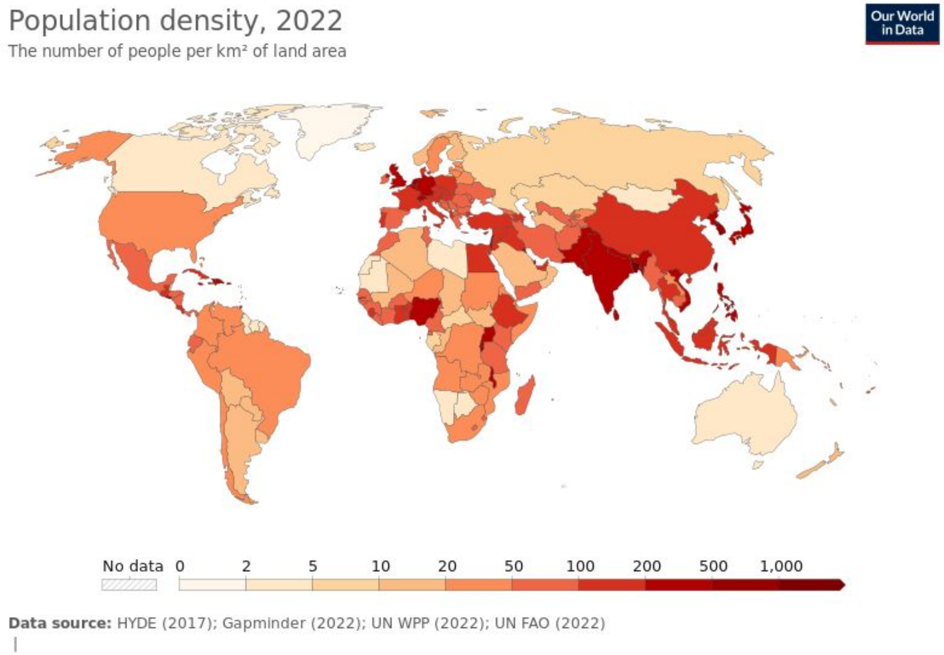
- States
- Counties and county equivalents
- County subdivisions
- Census tracts
- American Indian, Alaska Native, Native Hawaiian areas
- Tribal subdivisions
- Roads, rails, rivers
- School districts, etc.

Feb 07: What Is a Choropleth Map?

A choropleth map is a thematic map where areas are shaded or colored based on data values.

- Each region (e.g., state, county) is filled with a color corresponding to a data variable (e.g., population, unemployment rate).





Population density, 2022

The number of people per km² of land area

Data source: HYDE (2017); Gapminder (2022); UN WPP (2022); UN FAO (2022)

Let’s start executing Week3.ipynb together!

CORNELL UNIVERSITY FOUNDED A.D. 1865

PUBPOL 2130/ INFO 3130

Lab 4

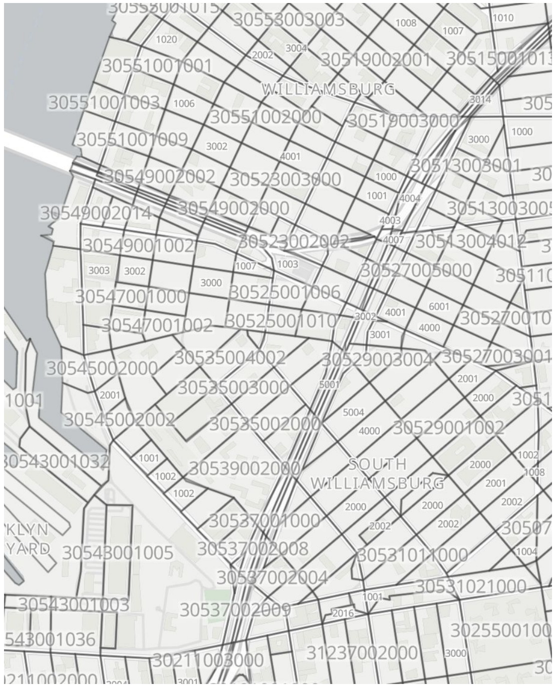
Feb 14: Announcements

- No new homework this week
- Seems you all did great on the test (and particularly on the Python problems)
- Python literacy is a learning objective -- understanding basic syntax is important
  - Spend time on the notebooks outside of class, and ask questions!
- We plan to offer a Python learning session some time next week if you need more support
  - Please fill out this google form: https://tinyurl.com/2130-poll

Feb 14: Census Blocks

Blocks:

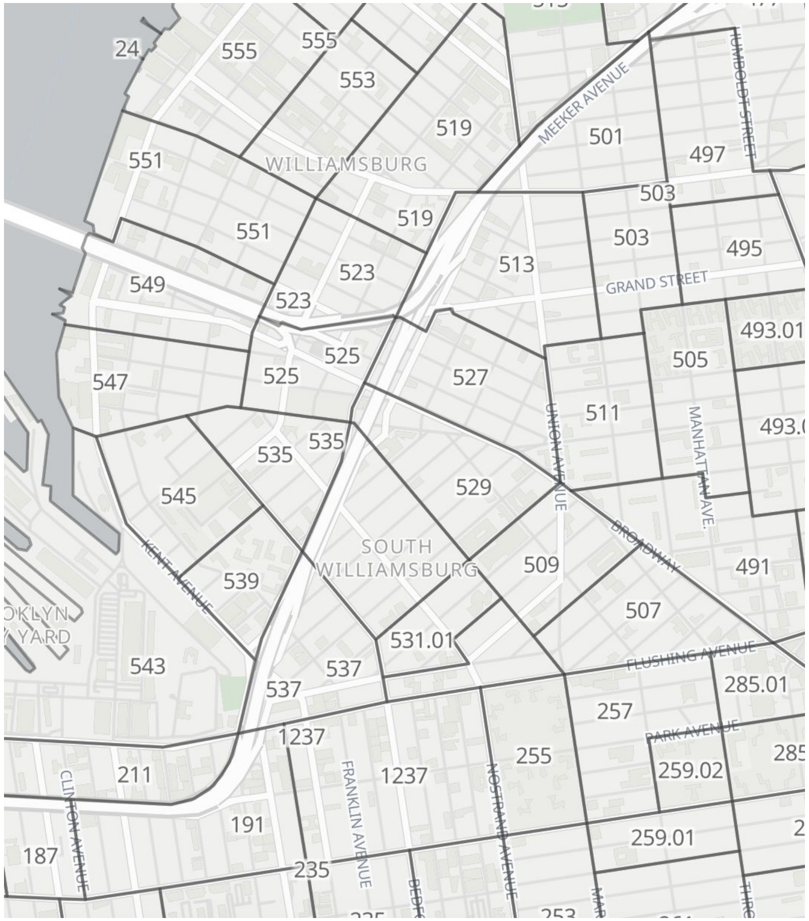
- Statistical areas with natural boundaries (e.g., roads)
- Cover the entire U.S.
- Smallest geographic unit for demographic data



Feb 14: Census Tracts

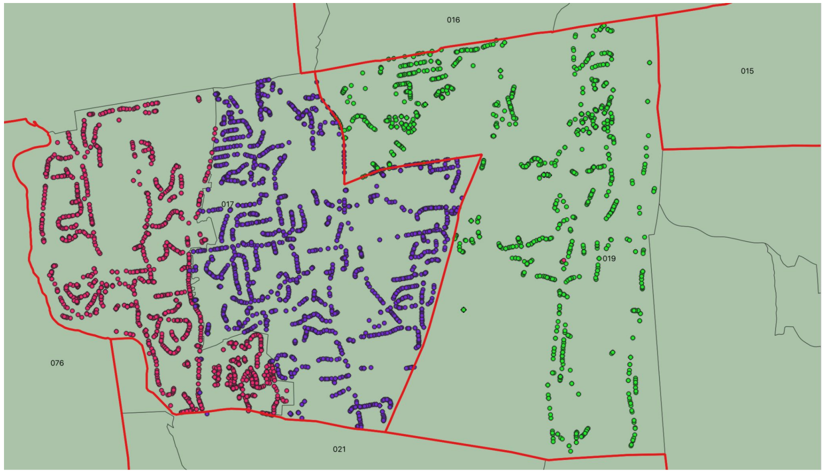
Tracts:

- Small, statistical subdivisions of counties
- Population between 1,200 and 8,000
- Spatial size varies widely



Feb 14: Precincts

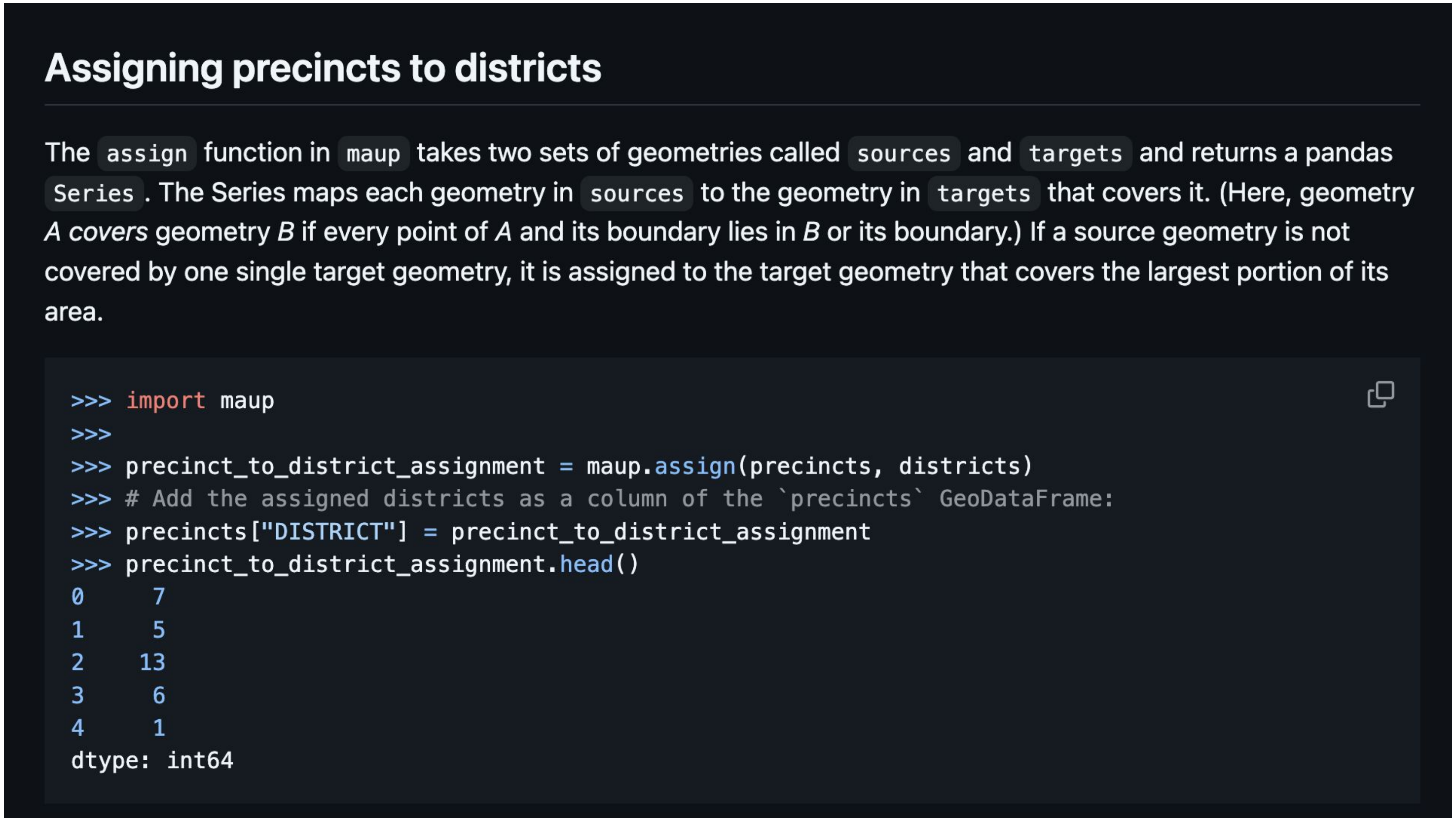
- Finest resolution of election data
- Not consistently maintained by states!
- mggg contains an open repository of precincts data



Feb 14: maup

- A geospatial toolkit for redistricting data
- Helpful for:
  - Aggregating from blocks to precincts
  - Disaggregating from precincts to blocks
  - “Prorating” data when there is no clean overlap

Feb 14: Assigning precincts to districts



Assigning precincts to districts

The `assign` function in `maup` takes two sets of geometries called `sources` and `targets` and returns a pandas `Series`. The Series maps each geometry in `sources` to the geometry in `targets` that covers it. (Here, geometry *A covers* geometry *B* if every point of *A* and its boundary lies in *B* or its boundary.) If a source geometry is not covered by one single target geometry, it is assigned to the target geometry that covers the largest portion of its area.

```
>>> import maup
>>>
>>> precinct_to_district_assignment = maup.assign(precincts, districts)
>>> # Add the assigned districts as a column of the `precincts` GeoDataFrame:
>>> precincts["DISTRICT"] = precinct_to_district_assignment
>>> precinct_to_district_assignment.head()
0     7
1     5
2    13
3     6
4     1
dtype: int64
```

Let’s start executing Week4.ipynb together!

PUBPOL 2130/ INFO 3130

Lab 5

Logistics!

- Created a slide to guide you through joins - please refer to it
- Today's notebook requires creation of a Google sheet, and map creation in flowmap.blue using this Google sheet; in CASE your sheet creation does not work, you can use these sheet IDs instead
  - yAnUt3bQcGpokOzteRQ2iGQUK3Lyw5S0OMwOOCa0wLQ
  - 1ffUAGYyzzPn3yY-0HehKnVsLayonauOLFtSPT3cPd0A
  - 1DoFxzh7_TKj2hbW7WV7fxbt0_MjjliKhAsGY7TDv4TE

Feb 21: Flowmaps

A flow map is a type of thematic map that visualizes movement or flow of objects, people, or data from one location to another.

Uses lines/arrows to show direction and magnitude of movement.

- **Lines/Arrows:** Represent movement direction.
- **Line Thickness/Color:** Can indicate volume/intensity of movement.
- **Nodes (Start/End Points):** Origin and destination of movement.
- **Base Map:** Provides spatial context (e.g., roads, cities, regions).

What we're using for Flowmaps: flowmap.blue

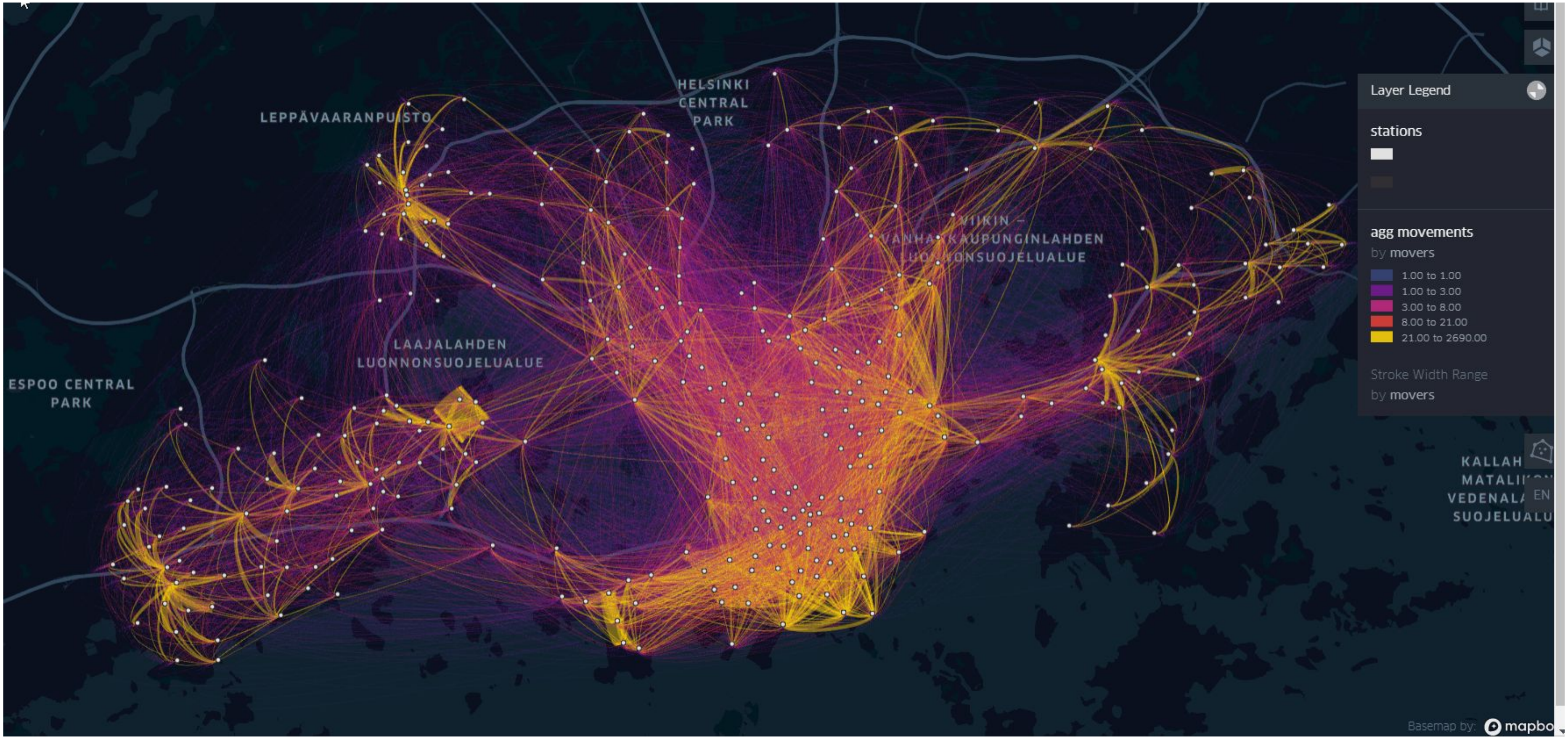
Feb 21: Flowmaps Applications

Flowmaps are commonly used to visualise these patterns

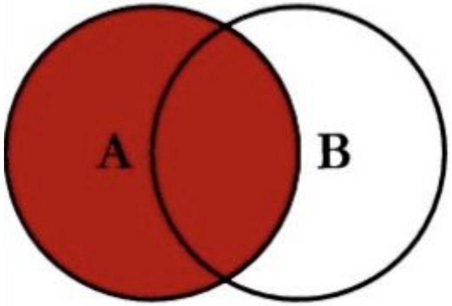
- **Migration Patterns:** People moving between cities or countries.
- **Trade Flows:** Import/export routes between regions.
- **Transportation & Traffic:** Airline routes, shipping lanes, or road traffic.
- **Energy Distribution:** Power grids, oil pipelines.
- **Internet Data Flow:** Digital connectivity between locations.

Feb 21: Flowmap example

Visualization of Bike Sharing System movements in Helsinki

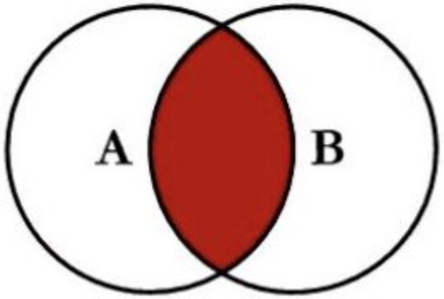


Feb 21: Joins (reference slide)



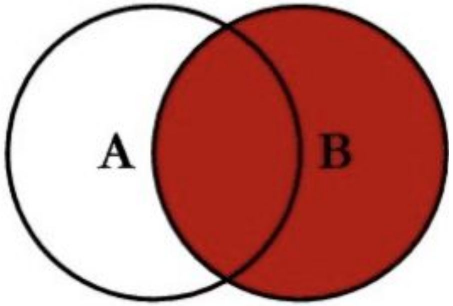
Keeps all elements in the left table, and common elements from the right table ONLY

Left Join



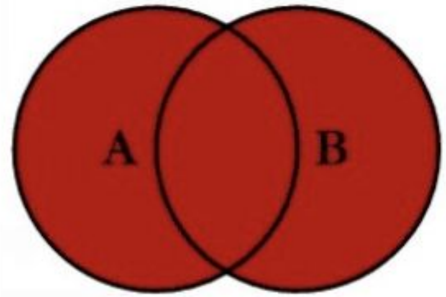
Keeps all elements common common to BOTH tables

Inner Join



Keeps all elements in the right table, and common elements from the left table ONLY

Right Join



Keeps ALL elements in from both tables.

Full Outer Join

df_joined = left_table.join(right_table, how = “outer”)

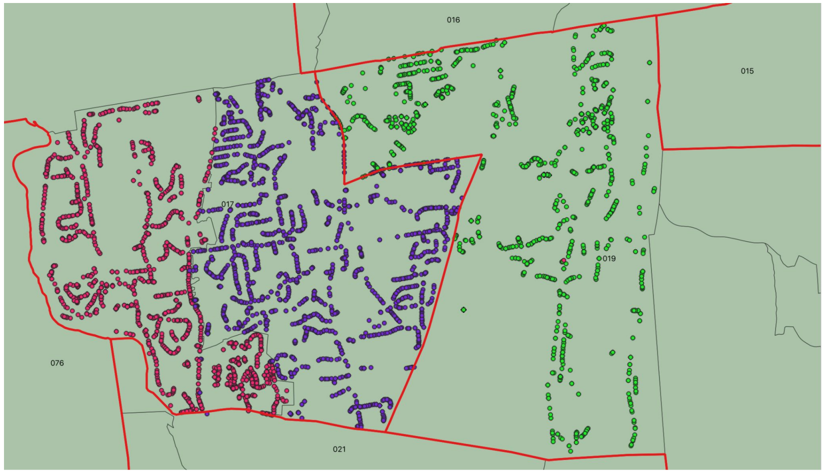
Let’s start executing Week5.ipynb together!

PUBPOL 2130/ INFO 3130

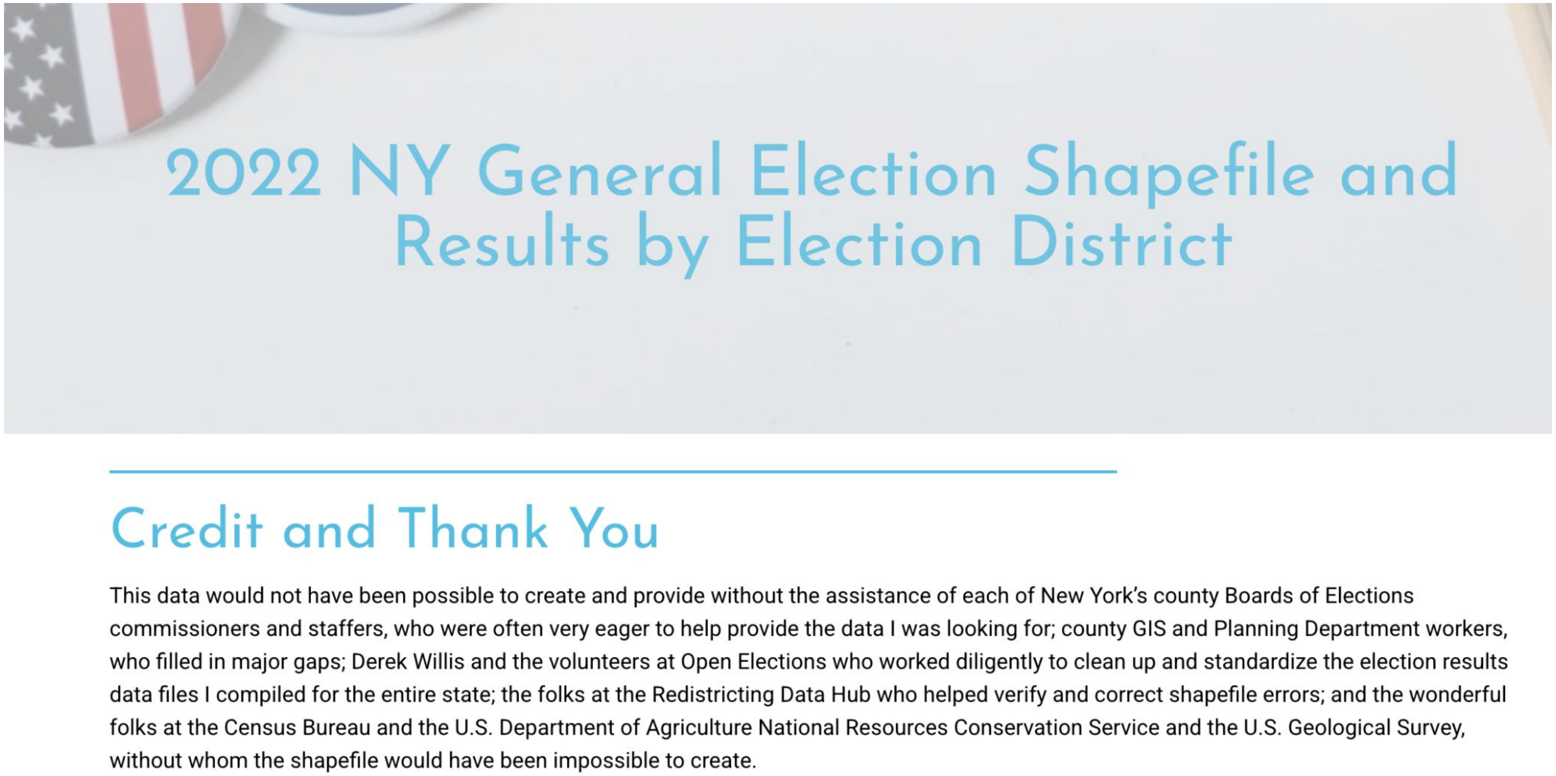
Lab 6

Feb 28: Precincts

- Finest resolution of election data
- Not consistently maintained by states!
- mggg contains an open repository of precincts data



Feb 28: Precincts



2022 NY General Election Shapefile and Results by Election District

Credit and Thank You

This data would not have been possible to create and provide without the assistance of each of New York's county Boards of Elections commissioners and staffers, who were often very eager to help provide the data I was looking for; county GIS and Planning Department workers, who filled in major gaps; Derek Willis and the volunteers at Open Elections who worked diligently to clean up and standardize the election results data files I compiled for the entire state; the folks at the Redistricting Data Hub who helped verify and correct shapefile errors; and the wonderful folks at the Census Bureau and the U.S. Department of Agriculture National Resources Conservation Service and the U.S. Geological Survey, without whom the shapefile would have been impossible to create.

Feb 28: maup

- A geospatial toolkit for redistricting data
- Potential uses:
    - Assigning precincts to districts,
    - Aggregating block data to precincts,
    - Disaggregating data from precincts down to blocks,
    - Prorating data when units do not nest neatly, and
    - Fixing topological issues, overlaps, and gaps

Feb 28: maup.assign()

```
blocks_to_precincts_assignment = maup.assign(blocks, pcts)
```

- Takes in two sets of geometries
- Returns a Pandas series with the assignments
- Use cases:
  - Assign blocks to precincts
  - Assign precincts to districts

Feb 28: maup.prorate()

```
maup.prorate(overlapping_blocks, data_to_prorate, weights)
```

- Problem:
  - You have precincts with some election results data
  - You also have different precincts (e.g., redistricting)
- We can **prorate** data from the old precincts to the new precincts by weighting the data proportional to the overlapping population.
  - Disaggregate to Census blocks
  - Reaggregate to new precincts

Let’s start executing Week6.ipynb together!

PUBPOL 2130/ INFO 3130

Lab 7

Announcements!

Weekly homework assignments:

- Homework will reflect the following submission timelines in Gradescope going forward
  - 7 days till submission deadline
  - +4 slip days
- i.e. you will still have up to 11 days to submit your homework!

Mar 07: PUMS Data

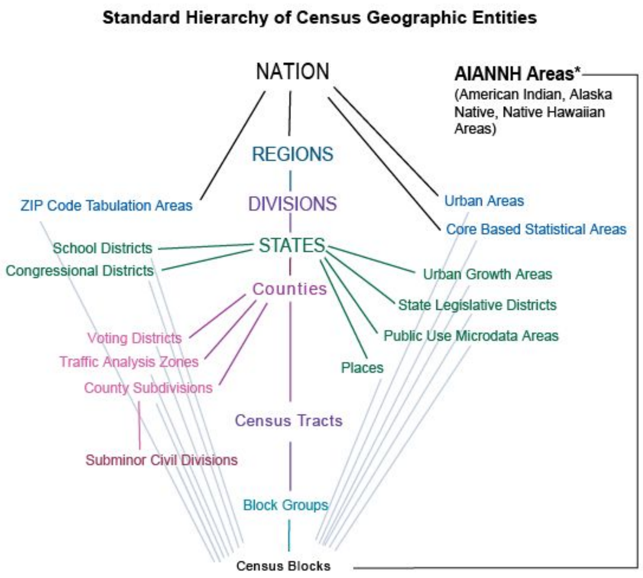
PUMS = Public Use Microdata Sample

- Provided by the Census Bureau
- Individual or household level information
    - Age
    - Race
    - Gender
    - Income
    - Employment
    - Housing variables

Mar 07: PUMS Data

PUMS = Public Use Microdata Sample

- What makes it "microdata"
  - ACS provides data at a Block Group or Tract level (most granular geography)
  - PUMS provides data at
    - An individual level, OR
    - A household level
    - Geographic granularity: PUMAs

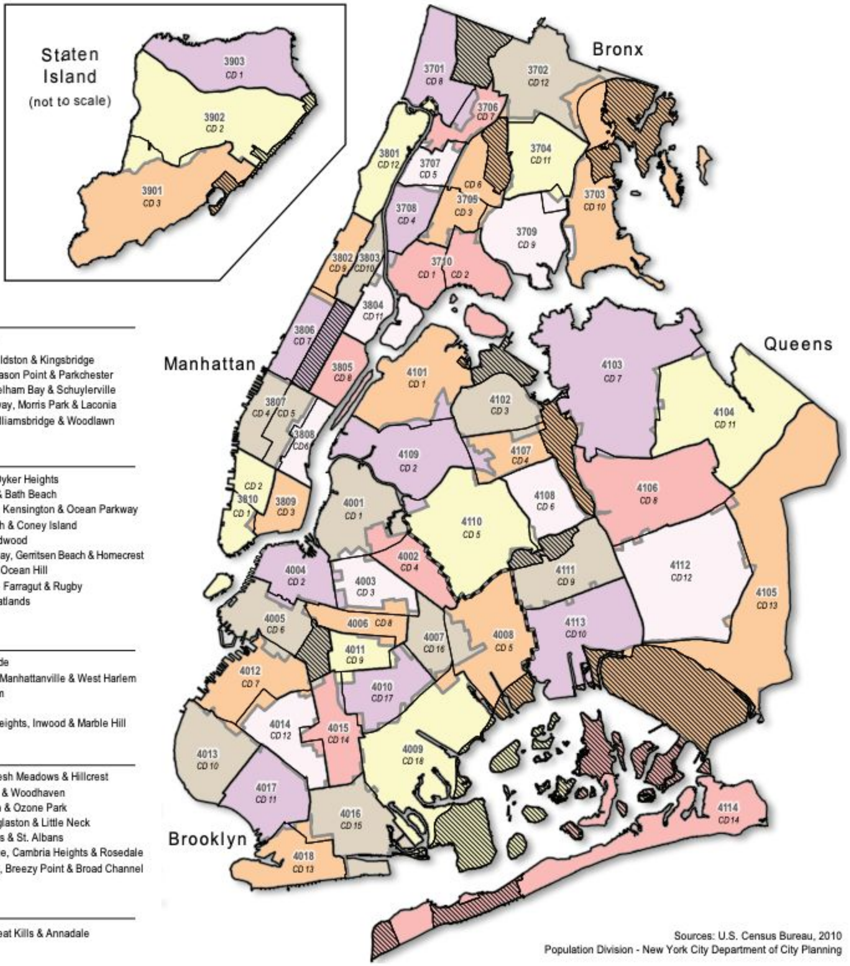


Mar 07: PUMS Data

New York City PUMAs and Community Districts

Public Use Microdata Areas (PUMAs) approximate NYC Community Districts (CDs).

- 3702 PUMAs
- CD 12 Community District boundaries
- ▨ Joint Interest Areas (JIAs) *e.g. parks and airports*



Sources: U.S. Census Bureau, 2010
Population Division - New York City Department of City Planning

Bronx

CD	PUMA	PUMA Name	CD	PUMA	PUMA Name
1 & 2	3710	Hunts Point, Longwood & Melrose	*8*	3701	Riverdale, Fieldston & Kingsbridge
3 & 6	3705	Belmont, Crotona Park East & East Tremont	*9*	3709	Castle Hill, Clason Point & Parkchester
4	3708	Concourse, Highbridge & Mount Eden	*10*	3703	Co-op City, Pelham Bay & Schuylerville
5	3707	Morris Heights, Fordham South & Mount Hope	*11*	3704	Pelham Parkway, Morris Park & Laconia
7	3706	Bedford Park, Fordham North & Norwood	*12*	3702	Wakefield, Williamsbridge & Woodlawn

Brooklyn

1	4001	Greenpoint & Williamsburg	*10*	4013	Bay Ridge & Dyker Heights
2	4004	Brooklyn Heights & Fort Greene	*11*	4017	Bensonhurst & Bath Beach
3	4003	Bedford-Stuyvesant	*12*	4014	Borough Park, Kensington & Ocean Parkway
4	4002	Bushwick	*13*	4018	Brighton Beach & Coney Island
5	4008	East New York & Starrett City	*14*	4015	Flatbush & Midwood
6	4005	Park Slope, Carroll Gardens & Red Hook	*15*	4016	Sheepshead Bay, Gerritsen Beach & Homecrest
7	4012	Sunset Park & Windsor Terrace	*16*	4007	Brownsville & Ocean Hill
8	4006	Crown Heights North & Prospect Heights	*17*	4010	East Flatbush, Farragut & Rugby
9	4011	Crown Heights So., Prospect Lefferts & Wingate	*18*	4009	Canarsie & Flatlands

Manhattan

1 & 2	3810	Battery Park City, Greenwich Village & Soho	*8*	3805	Upper East Side
3	3809	Chinatown & Lower East Side	*9*	3802	Hamilton Hts, Manhattanville & West Harlem
4 & 5	3807	Chelsea, Clinton & Midtown Business District	*10*	3803	Central Harlem
6	3808	Murray Hill, Gramercy & Stuyvesant Town	*11*	3804	East Harlem
7	3806	Upper West Side & West Side	*12*	3801	Washington Heights, Inwood & Marble Hill

Queens

1	4101	Astoria & Long Island City	*8*	4106	Briarwood, Fresh Meadows & Hillcrest
2	4109	Sunnyside & Woodside	*9*	4111	Richmond Hill & Woodhaven
3	4102	Jackson Heights & North Corona	*10*	4113	Howard Beach & Ozone Park
4	4107	Elmhurst & South Corona	*11*	4104	Bayside, Douglaston & Little Neck
5	4110	Ridgewood, Glendale & Middle Village	*12*	4112	Jamaica, Hollis & St. Albans
6	4108	Forest Hills & Rego Park	*13*	4105	Queens Village, Cambria Heights & Rosedale
7	4103	Flushing, Murray Hill & Whitestone	*14*	4114	Far Rockaway, Breezy Point & Broad Channel

Staten Island

1	3903	Port Richmond, Stapleton & Mariner's Harbor	*3*	3901	Tottenville, Great Kills & Annadale
2	3902	New Springville & South Beach			

Mar 07: PUMS Data

- Where to access PUMS data
  - Census Bureau: https://www.census.gov/programs-surveys/acs/microdata/access.html
  - IPUMS: https://usa.ipums.org/usa/
  - Census Bureau data portal: https://data.census.gov/cedsci/

Let’s start executing Week7.ipynb together!

PUBPOL 2130/ INFO 3130

Lab 8

Announcements!

Prelim & Final Exam Dates!

- Next prelim is March 25th
- Potentially optional third exam (announced on Tuesday)
- Final deliverable due Wed, May 14th @12:00 PM
- HW6 Warmup Question Change! (See Gradescope)

Reminder:

- Today’s data is **sensitive!** Do not distribute it – you will need to attest that you have destroyed it at the end of the semester.

Mar 14: OPTN Data

- OPTN = Organ Procurement & Transplantation Network

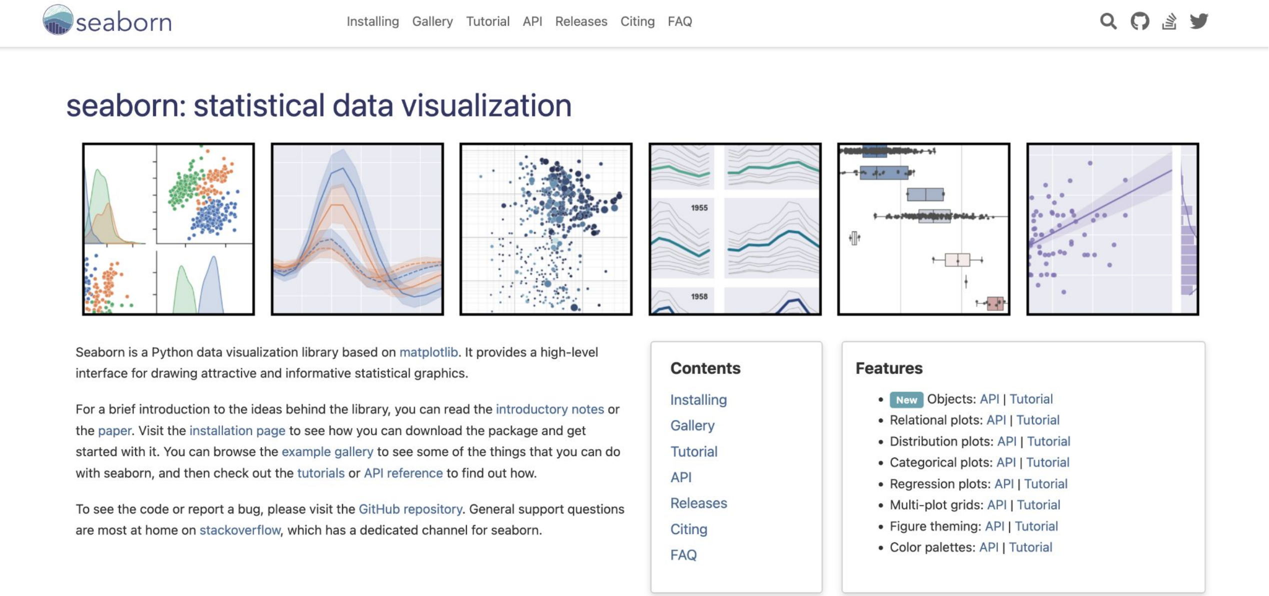
- Data is available here

- Contains pre- and post-transplant information on:
  - Waiting list candidates
  - donor/recipient matches
  - Deceased and living donors

- We use STAR (Standard Transplant Analysis and Research) files

Mar 14: Seaborn

- Today’s notebook also uses **seaborn**



seaborn: statistical data visualization

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

For a brief introduction to the ideas behind the library, you can read the introductory notes or the paper. Visit the installation page to see how you can download the package and get started with it. You can browse the example gallery to see some of the things that you can do with seaborn, and then check out the tutorials or API reference to find out how.

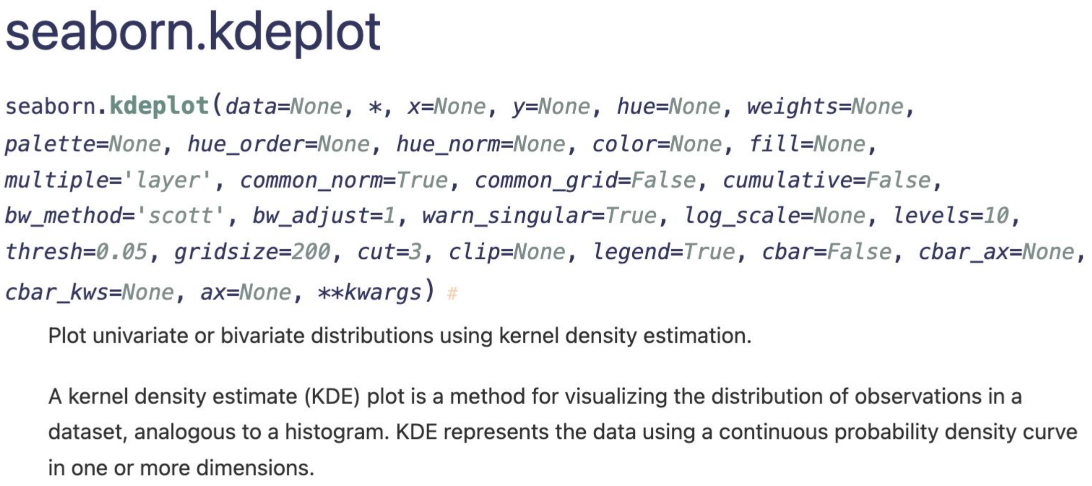
To see the code or report a bug, please visit the GitHub repository. General support questions are most at home on stackoverflow, which has a dedicated channel for seaborn.

Contents

Features

- New Objects: API | Tutorial
- Relational plots: API | Tutorial
- Distribution plots: API | Tutorial
- Categorical plots: API | Tutorial
- Regression plots: API | Tutorial
- Multi-plot grids: API | Tutorial
- Figure theming: API | Tutorial
- Color palettes: API | Tutorial

Mar 14: Seaborn: kdeplot

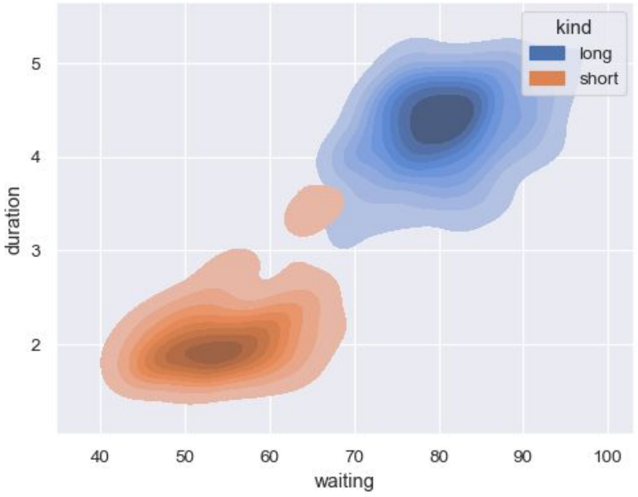


seaborn.kdeplot

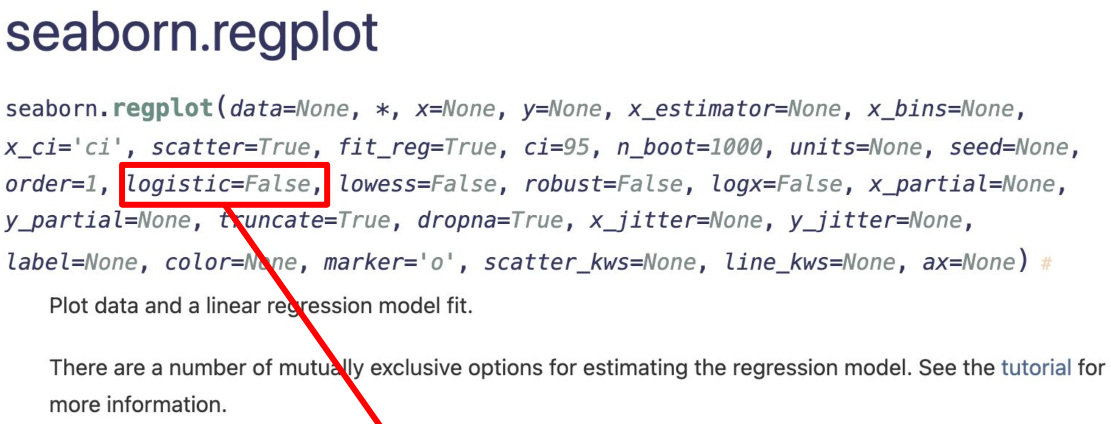
```
seaborn.kdeplot(data=None, *, x=None, y=None, hue=None, weights=None,
palette=None, hue_order=None, hue_norm=None, color=None, fill=None,
multiple='layer', common_norm=True, common_grid=False, cumulative=False,
bw_method='scott', bw_adjust=1, warn_singular=True, log_scale=None, levels=10,
thresh=0.05, gridsize=200, cut=3, clip=None, legend=True, cbar=False, cbar_ax=None,
cbar_kws=None, ax=None, **kwargs) #
```

Plot univariate or bivariate distributions using kernel density estimation.

A kernel density estimate (KDE) plot is a method for visualizing the distribution of observations in a dataset, analogous to a histogram. KDE represents the data using a continuous probability density curve in one or more dimensions.



Mar 14: Seaborn: kdeplot

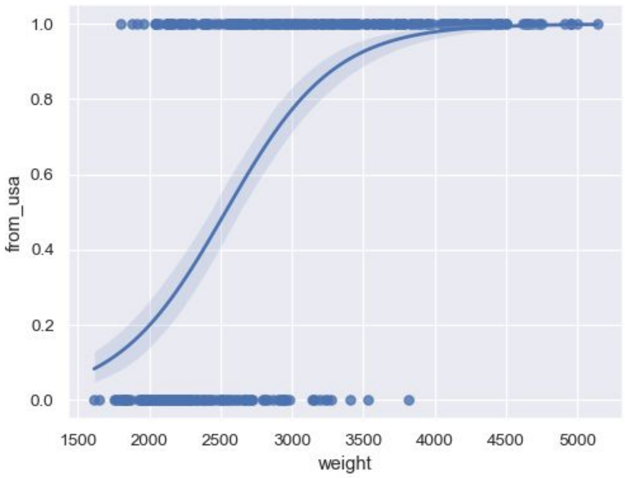


seaborn.regplot

```
seaborn.regplot(data=None, *, x=None, y=None, x_estimator=None, x_bins=None,
x_ci='ci', scatter=True, fit_reg=True, ci=95, n_boot=1000, units=None, seed=None,
order=1, logistic=False, lowess=False, robust=False, logx=False, x_partial=None,
y_partial=None, truncate=True, dropna=True, x_jitter=None, y_jitter=None,
label=None, color=None, marker='o', scatter_kws=None, line_kws=None, ax=None) #
```

Plot data and a linear regression model fit.

There are a number of mutually exclusive options for estimating the regression model. See the tutorial for more information.



For binary outcomes

Let’s start executing Week8.ipynb together!