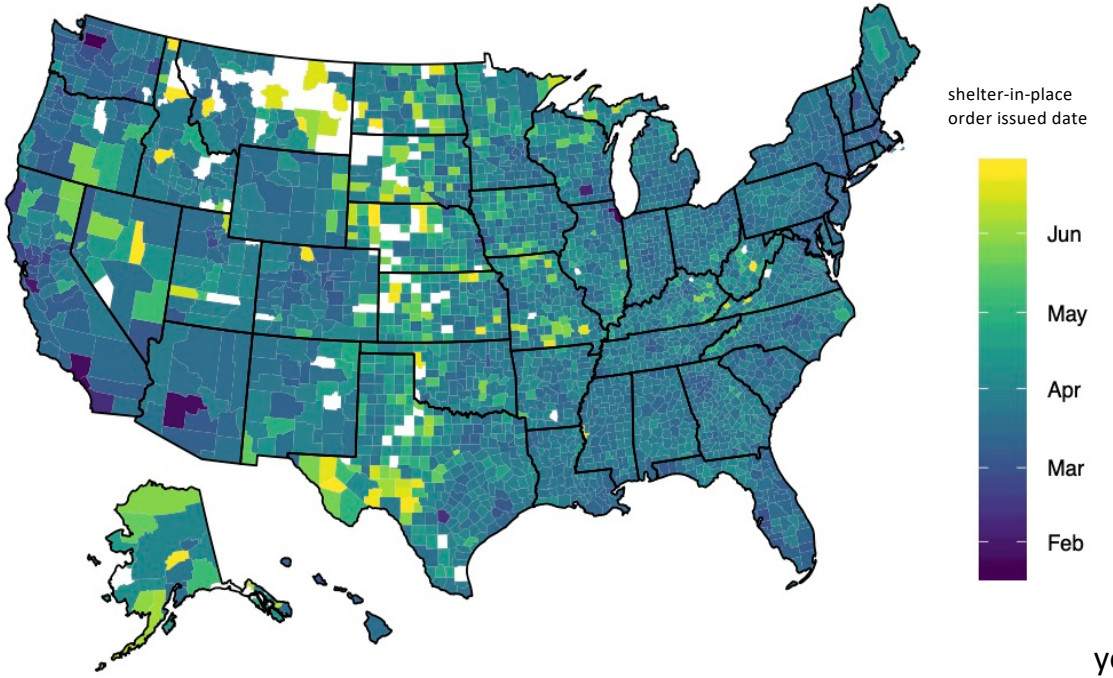
S&DS 177
YData: COVID-19 Behavioral Impacts



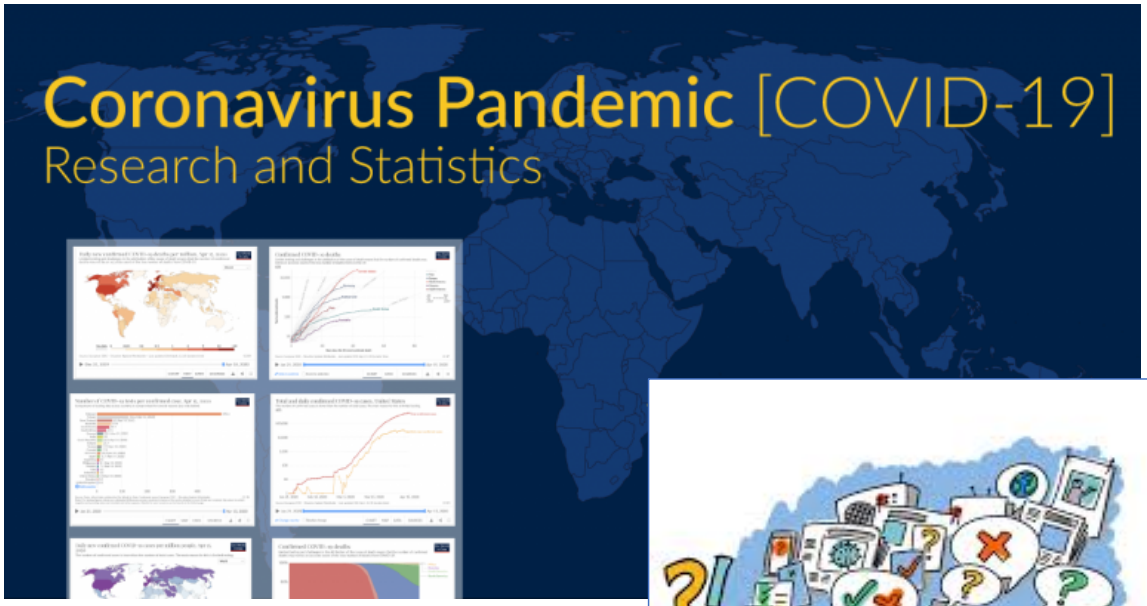
Youpei Yan
youpei.yan@yale.edu

Lecture 2 (Feb 11, Thursday 9:30 – 12:00)

Overview:

Process data to extract useful information:

- Data Sorting & Selection
- Add New Variables
- Group Data by Column Values
- Visualization



Lecture 2 (Feb 11, Thursday 9:30 – 12:00) 1. Data Sorting & Selection

Let's first import data using read_table('path')

We define a dataset named nytdata,
And assign the table to it with "=" mark

We then call it by typing the table name: nytdata

Jupyter Notebook will show the first several rows of this table to give us an idea:

- What variables do we have
- How they look like
- How many rows in total

```
from datascience import *

nytdata = Table.read_table('YData_SDS177/nyt_cases_name.csv')
nytdata
```

geoid	county	state	cases	deaths	date	dwell_time
1001	Autauga	AL	0	0	01jan2020	929.958
1003	Baldwin	AL	0	0	01jan2020	857.864
1005	Barbour	AL	0	0	01jan2020	788.396
1007	Bibb	AL	0	0	01jan2020	913.771
1009	Blount	AL	0	0	01jan2020	956.059
1011	Bullock	AL	0	0	01jan2020	710.707
1013	Butler	AL	0	0	01jan2020	813.134
1015	Calhoun	AL	0	0	01jan2020	837.673
1017	Chambers	AL	0	0	01jan2020	875.582
1019	Cherokee	AL	0	0	01jan2020	820.602

... (456278 rows omitted)

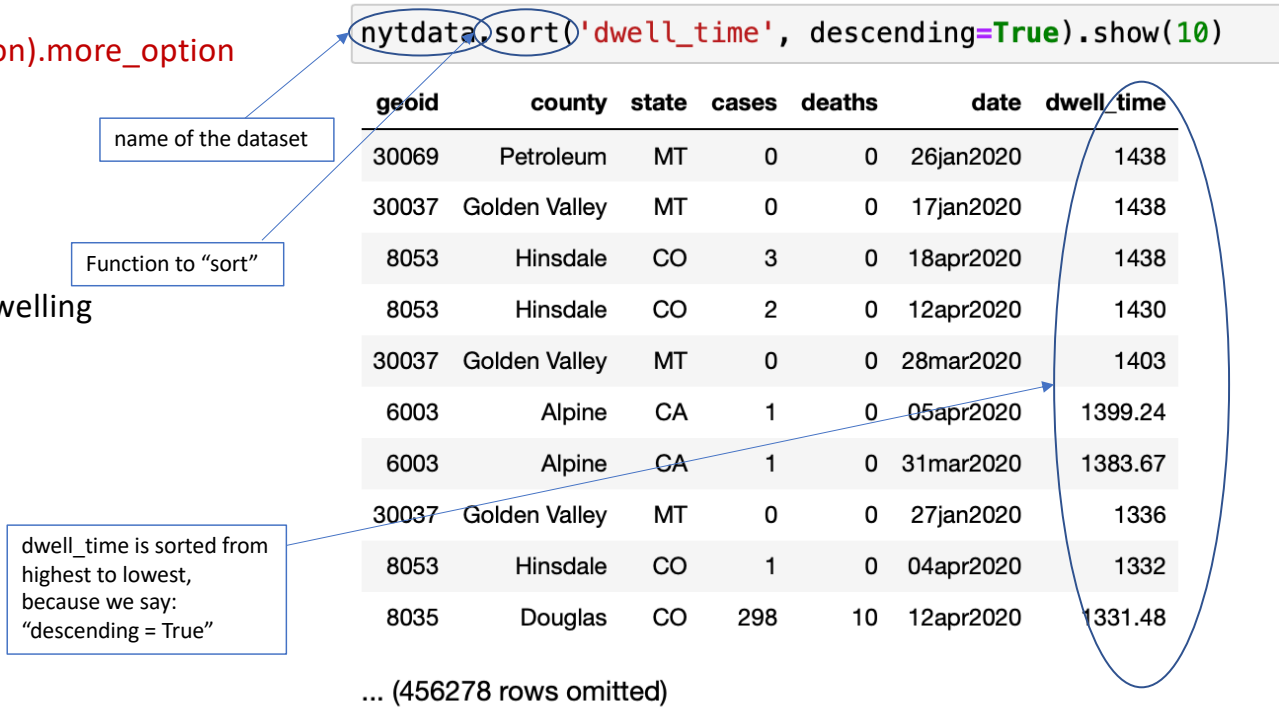
Lecture 2 (Feb 11, Thursday 9:30 – 12:00) 1. Data Sorting & Selection

- The format of using functions are the same in Python, and they usually look like this:

Tablename.function('variable', option).more_option

- Example of sorting:

We sort the dataset nytdata based on the value of "dwell_time", which is the average time people dwelling at home for a county in a day.



geoid	county	state	cases	deaths	date	dwell_time
30069	Petroleum	MT	0	0	26jan2020	1438
30037	Golden Valley	MT	0	0	17jan2020	1438
8053	Hinsdale	CO	3	0	18apr2020	1438
8053	Hinsdale	CO	2	0	12apr2020	1430
30037	Golden Valley	MT	0	0	28mar2020	1403
6003	Alpine	CA	1	0	05apr2020	1399.24
6003	Alpine	CA	1	0	31mar2020	1383.67
30037	Golden Valley	MT	0	0	27jan2020	1336
8053	Hinsdale	CO	1	0	04apr2020	1332
8035	Douglas	CO	298	10	12apr2020	1331.48

... (456278 rows omitted)

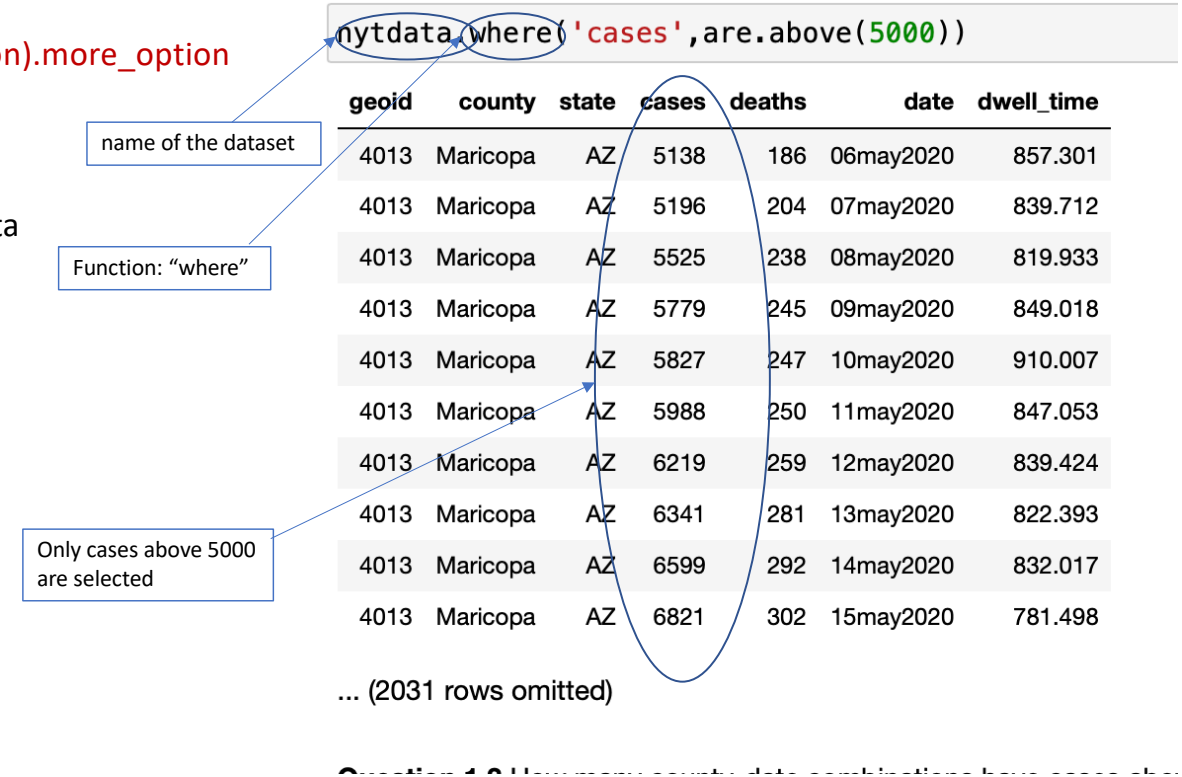
Lecture 2 (Feb 11, Thursday 9:30 – 12:00) 1. Data Sorting & Selection

- The format of using functions are the same in Python, and they usually look like this:

Table name.function('variable', option).more_option

- Example of selecting:

We select rows of the dataset nytdata based on the value of "cases".



geoid	county	state	cases	deaths	date	dwell_time
4013	Maricopa	AZ	5138	186	06may2020	857.301
4013	Maricopa	AZ	5196	204	07may2020	839.712
4013	Maricopa	AZ	5525	238	08may2020	819.933
4013	Maricopa	AZ	5779	245	09may2020	849.018
4013	Maricopa	AZ	5827	247	10may2020	910.007
4013	Maricopa	AZ	5988	250	11may2020	847.053
4013	Maricopa	AZ	6219	259	12may2020	839.424
4013	Maricopa	AZ	6341	281	13may2020	822.393
4013	Maricopa	AZ	6599	292	14may2020	832.017
4013	Maricopa	AZ	6821	302	15may2020	781.498

... (2031 rows omitted)

- Can we combine the above two functions in one line? Yes! Because after sorting or selecting, we still get a ”table”, and we can further add ”.function()” behind it.

```
nytdata.where('cases',are.above(5000)).sort('dwell_time',descending=True)
```

A table

A function to change the table

A table

A function to change the table

- Now, let us try to select county-day combination with an average dwell_time above 1000 minutes, then sort it by descending “cases”, which county becomes the first?

Lecture 2 (Feb 11, Thursday 9:30 – 12:00) 2. Add New Variables

The function to add a new column:
Tablename.with_columns(
'var name', value
)

```
nyt_cases = nytdata.with_columns(
        'log transformation of cases', np.log(nytdata.column('cases')+1)
)
nyt_cases.sort('log transformation of cases', descending=True)
```

New variable name

We set the value of it to be log(cases + 1)

geoid	county	state	cases	deaths	date	dwell_time	log transformation of cases
36061	New York	NY	204111	20795	26may2020	590.555	12.2264
36061	New York	NY	203569	20740	25may2020	604.917	12.2238
36061	New York	NY	202931	20697	24may2020	596.714	12.2206
36061	New York	NY	202062	20621	23may2020	647.159	12.2163
36061	New York	NY	201298	20569	22may2020	643.844	12.2125
36061	New York	NY	200507	20491	21may2020	689.287	12.2086
36061	New York	NY	199392	20422	20may2020	726.176	12.203
36061	New York	NY	198710	20376	19may2020	738.072	12.1996
36061	New York	NY	198114	20298	18may2020	723.077	12.1966
36061	New York	NY	197486	20214	17may2020	739.853	12.1934

... (456278 rows omitted)

Lecture 2 (Feb 11, Thursday 9:30 – 12:00) 3. Group Data

- Although the case and death reports for each county and each day show us the details, we sometimes want to see a summary of reports to quickly identify, say, which state(s) have a high case reports, which date period were the epidemic reached to the peak, etc. To do so, we would love to use a method to sum data by group.

The function to find group mean/max/min is:
Tablename.group('var name', option)

- Also try the following

nytdata.group('state', max)
nytdata.group('state', min)
nytdata.group('state', average)

nytdata.group('date',max)
nytdata.group('date',min)
nytdata.group('date',np.mean)

All the variables in the table are group means of the original values. For instance, 42.4839 is the average case number in AL over the study period.

```
# find the state mean
state_group = nytdata.group('state', np.mean)
state_group
```

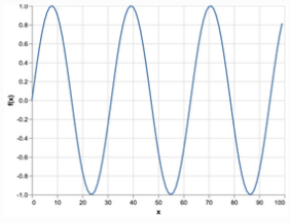
state	geoid mean	county mean	cases mean	deaths mean	date mean	dwell_time mean
AL	1067		42.4839	1.57295	1.58416e+09	701.909
AR	5075		15.199	0.320272	1.58416e+09	664.021
AZ	4013.87		208.6	9.23537	1.58416e+09	572.695
CA	6058		335.603	12.9089	1.58416e+09	718.563
CO	8062.23		85.9373	4.30251	1.58416e+09	551.878
CT	9008		1171.04	94.8656	1.58416e+09	688.028
DC	11001		1673.64	79.6054	1.58416e+09	519.904
DE	10003		569.034	19.1973	1.58416e+09	659.63
FL	12067.9		185.725	6.86506	1.58416e+09	638.372
GA	13161.5		56.114	2.47166	1.58416e+09	683.399

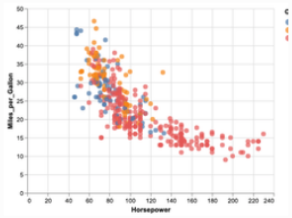
... (39 rows omitted)

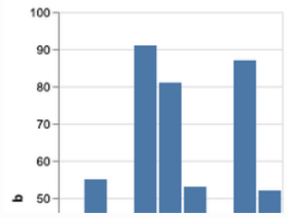
Lecture 2 (Feb 11, Thursday 9:30 – 12:00)4. Visualization

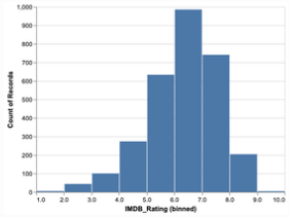
```
# this code makes it so that figures appear in Jupyter notebooks, do not remove it!
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

- Creating figures based on the dataset is always a more straightforward way than reading numbers.
- The above lines are useful for us to make figures in Jupyter Notebooks.
- What types of statistical figures have you seen before?
  - Scatter
  - Lines
  - Bar
  - Histogram
  - Etc.


Simple Line Chart

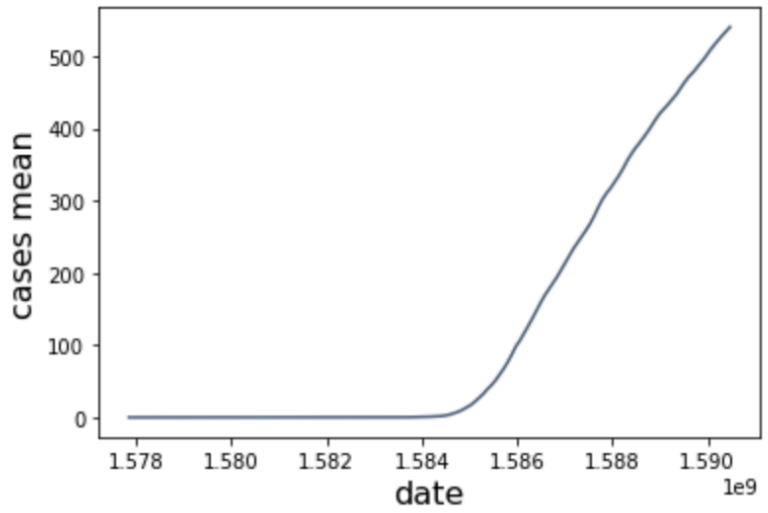

Simple Scatter Plot


Simple Bar Chart


Simple Histogram

The function to plot a line relationship:
Tablename.plot(
'var on x-axis', 'var on y-axis'
)

```
date_group.plot('date','cases mean')
```



- The date is converted to numerical values from strings, yet the default conversion is to nanoseconds. That is why we see such huge numbers on the x-axis.

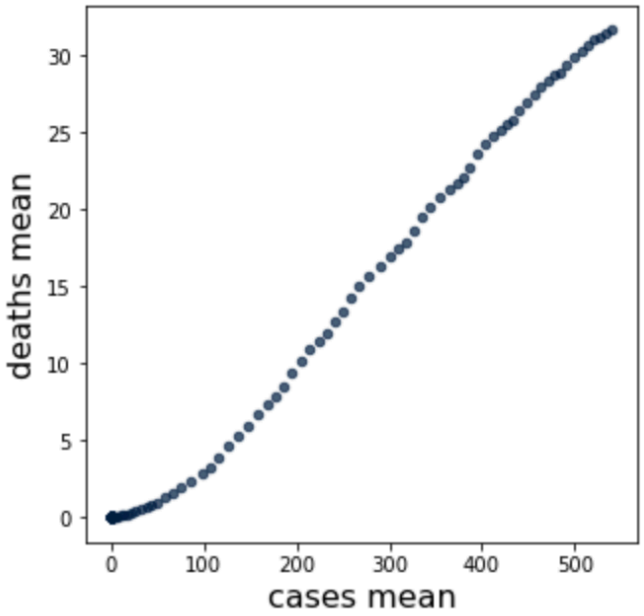
- The unit of the axes matter a lot. Current, we would know that in the middle of the study period, the case number is increasing, but it's hard to tell when.

(We'll deal with this issue with the correct format next class, now we just need to know that they are well-ordered.)

Lecture 2 (Feb 11, Thursday 9:30 – 12:00)4. Visualization

The function to plot scatter:
Tablename.scatter(
'var on x-axis', 'var on y-axis'
)

```
date_group.scatter('cases mean','deaths mean')
```



- What can we say from this figure?
- Do you think this is a good prediction of deaths?
- Why or why not?

Lecture 2 (Feb 11, Thursday 9:30 – 12:00)4. Visualization

The function to make a horizontal bar plot:
Tablename.barh(
'var on y-axis', 'var on x-axis'
)

- What to do to make a vertical bar plot?
- Try sorting the dwell_time mean first before plotting.

The function to plot histogram:
Tablename.hist('var')

- Why just one variable here? It's because histogram is a diagram consisting of rectangles whose area is proportional to the frequency of a variable and whose width is equal to the class interval.
- Try to plot the histogram of "dwell_time mean".

```
state_group.barh('state','dwell_time mean')
```