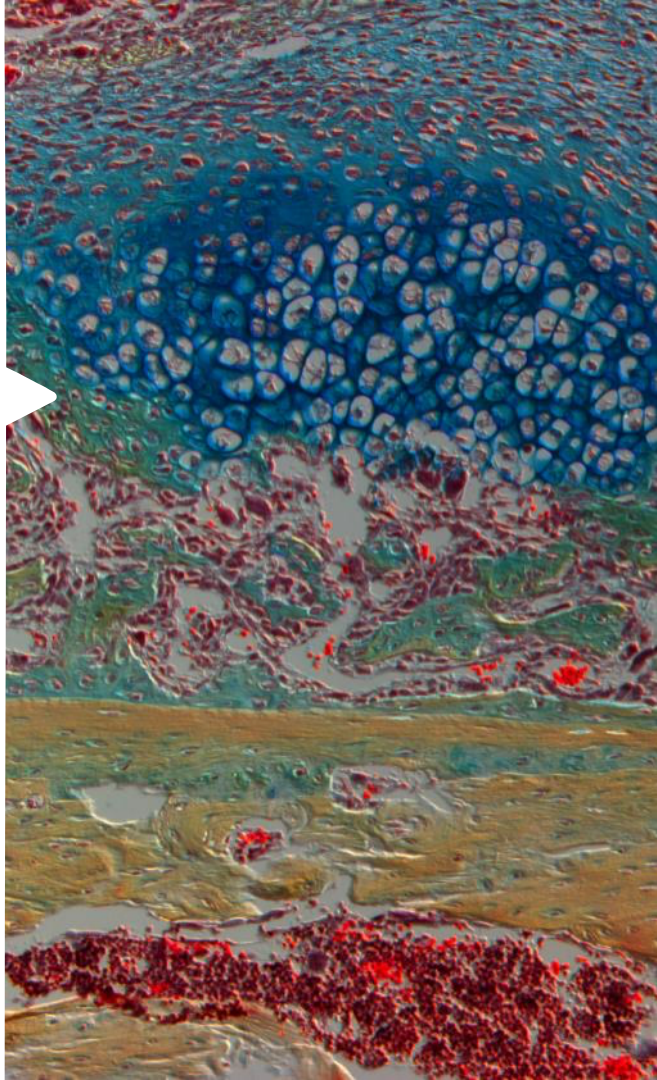
NYU Langone Health

Pandas Basics

Write Subtitle In This Area

Division Name



Todays Lecture

- Presentation (30 min)
  - Pandas
- Poll / Quiz on discussed Pandas Topics (10 min)
- Break / Questions (10 min)
- Worked Example (50 min)
  - I will write code to analyze a dataset
  - You will then reproduce the steps in breakout rooms

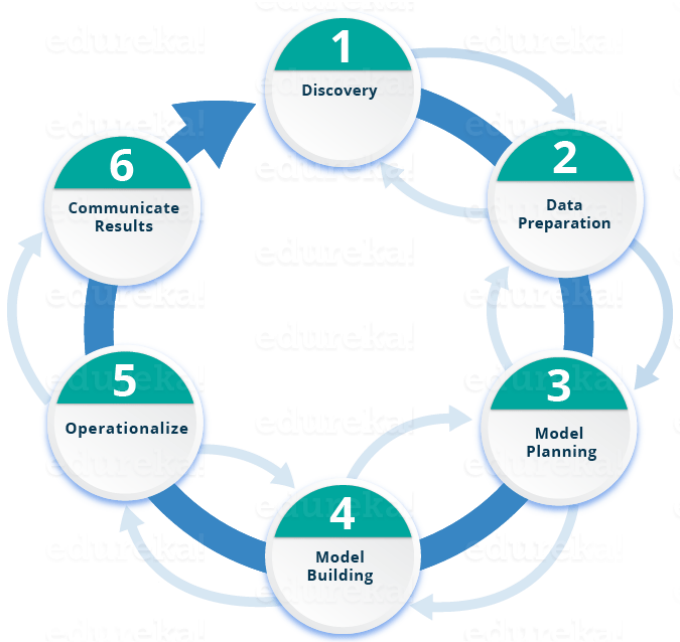
- Using Pandas to generate plots (time dependent)

Pandas - History

- Created by Wes McKinney in 2008 while working for AQR Capital Management
  - He convinced the company to make the project open source
- Chang She joined the project in 2012 and was the second major contributor
- Pandas signed onto NumFOCUS, a nonprofit charity, in 2015
- Pandas remains a go to package for working tabular data
  - Many packages integrate with pandas
    - Seaborn for visualization
    - Statsmodels for statistical models
    - Sklearn - machine learning
    - Many more

Pandas – Overview

- Data analysis
  - Cleaning
  - Transforming
  - Visualizing
- Pandas Documentation
  - https://pandas.pydata.org/pandas-docs/stable/
  - Google is my go-to method
    - Search ‘pandas <function>’
    - Choose the pandas.pydata.org link



https://www.edureka.co/blog/what-is-data-science/

NYU Langone Health

Pandas – Installation / Import

- The source code is available on GitHub (link)
- In almost all cases you would use a package manager to install Pandas
  - Python package index

    ```
    pip install pandas
    ```

  - Conda

    ```
    conda install pandas
    ```

  - Pandas is a default package on Google Colab, so it does not need to be installed
- Importing Pandas

  ```
  import pandas as pd
  ```

  - ‘pd’ is a common short name used for pandas so that ‘pd’ can be used instead of typing ‘pandas’

Example Dataset

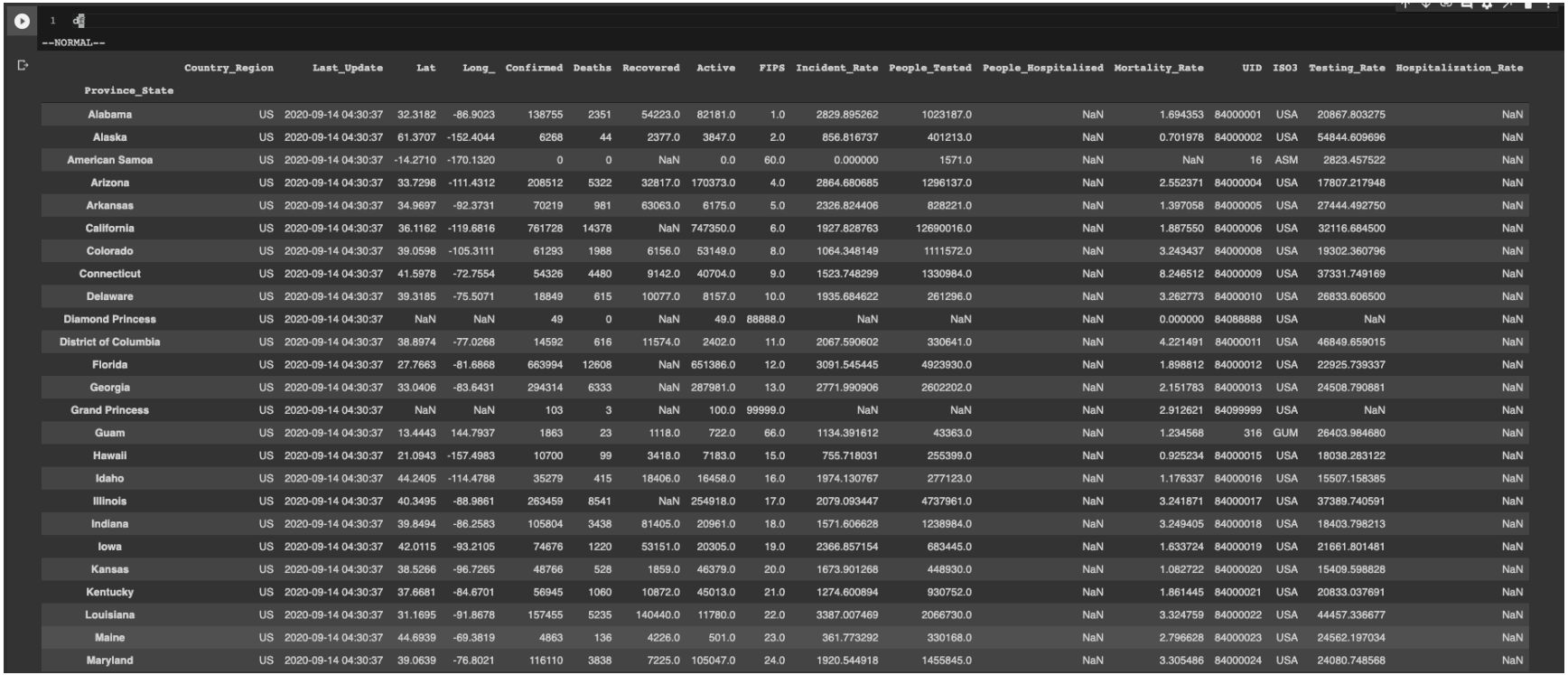
- JHU CSSE COVID-19 Dataset
  - https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data
  - https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_daily_reports_us/09-13-2020.csv
- For today we will be looking at the daily reports for the US
- On Thursday we will use the time series dataset for the US

- My code for loading the example dataset:

```
df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_daily_reports_us/09-13-2020.csv')
df = df.set_index('Province_State')
```


NYU Langone Health

Example Dataset



--NORMAL--

Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	FIPS	Incident_Rate	People_Tested	People_Hospitalized	Mortality_Rate	UID	ISO3	Testing_Rate	Hospitalization_Rate
Alabama	US	2020-09-14 04:30:37	32.3182	-86.9023	138755	2351	54223.0	82181.0	1.0	2829.895262	1023187.0	NaN	1.694353	84000001	USA	20867.803275	NaN
Alaska	US	2020-09-14 04:30:37	61.3707	-152.4044	6268	44	2377.0	3847.0	2.0	856.816737	401213.0	NaN	0.701978	84000002	USA	54844.609696	NaN
American Samoa	US	2020-09-14 04:30:37	-14.2710	-170.1320	0	0	NaN	0.0	60.0	0.000000	1571.0	NaN	NaN	16	ASM	2823.457522	NaN
Arizona	US	2020-09-14 04:30:37	33.7298	-111.4312	208512	5322	32817.0	170373.0	4.0	2864.680685	1296137.0	NaN	2.552371	84000004	USA	17807.217948	NaN
Arkansas	US	2020-09-14 04:30:37	34.9697	-92.3731	70219	981	63063.0	6175.0	5.0	2326.824406	828221.0	NaN	1.397058	84000005	USA	27444.492750	NaN
California	US	2020-09-14 04:30:37	36.1162	-119.6816	761728	14378	NaN	747350.0	6.0	1927.828763	12690016.0	NaN	1.887550	84000006	USA	32116.684500	NaN
Colorado	US	2020-09-14 04:30:37	39.0598	-105.3111	61293	1988	6156.0	53149.0	8.0	1064.348149	1111572.0	NaN	3.243437	84000008	USA	19302.360796	NaN
Connecticut	US	2020-09-14 04:30:37	41.5978	-72.7554	54326	4480	9142.0	40704.0	9.0	1523.748299	1330984.0	NaN	8.246512	84000009	USA	37331.749169	NaN
Delaware	US	2020-09-14 04:30:37	39.3185	-75.5071	18849	615	10077.0	8157.0	10.0	1935.684622	261296.0	NaN	3.262773	84000010	USA	26833.606500	NaN
Diamond Princess	US	2020-09-14 04:30:37	NaN	NaN	49	0	NaN	49.0	88888.0	NaN	NaN	NaN	0.000000	84088888	USA	NaN	NaN
District of Columbia	US	2020-09-14 04:30:37	38.8974	-77.0268	14592	616	11574.0	2402.0	11.0	2067.590602	330641.0	NaN	4.221491	84000011	USA	46849.659015	NaN
Florida	US	2020-09-14 04:30:37	27.7663	-81.6868	663994	12608	NaN	651386.0	12.0	3091.545445	4923930.0	NaN	1.898812	84000012	USA	22925.739337	NaN
Georgia	US	2020-09-14 04:30:37	33.0406	-83.6431	294314	6333	NaN	287981.0	13.0	2771.990906	2602202.0	NaN	2.151783	84000013	USA	24508.790881	NaN
Grand Princess	US	2020-09-14 04:30:37	NaN	NaN	103	3	NaN	100.0	99999.0	NaN	NaN	NaN	2.912621	84099999	USA	NaN	NaN
Guam	US	2020-09-14 04:30:37	13.4443	144.7937	1863	23	1118.0	722.0	66.0	1134.391612	43363.0	NaN	1.234568	316	GUM	26403.984680	NaN
Hawaii	US	2020-09-14 04:30:37	21.0943	-157.4983	10700	99	3418.0	7183.0	15.0	755.718031	255399.0	NaN	0.925234	84000015	USA	18038.283122	NaN
Idaho	US	2020-09-14 04:30:37	44.2405	-114.4788	35279	415	18406.0	16458.0	16.0	1974.130767	277123.0	NaN	1.176337	84000016	USA	15507.158385	NaN
Illinois	US	2020-09-14 04:30:37	40.3495	-88.9861	263459	8541	NaN	254918.0	17.0	2079.093447	4737961.0	NaN	3.241871	84000017	USA	37389.740591	NaN
Indiana	US	2020-09-14 04:30:37	39.8494	-86.2583	105804	3438	81405.0	20961.0	18.0	1571.606628	1238984.0	NaN	3.249405	84000018	USA	18403.798213	NaN
Iowa	US	2020-09-14 04:30:37	42.0115	-93.2105	74676	1220	53151.0	20305.0	19.0	2366.857154	683445.0	NaN	1.633724	84000019	USA	21661.801481	NaN
Kansas	US	2020-09-14 04:30:37	38.5266	-96.7265	48766	528	1859.0	46379.0	20.0	1673.901268	448930.0	NaN	1.082722	84000020	USA	15409.598828	NaN
Kentucky	US	2020-09-14 04:30:37	37.6681	-84.6701	56945	1060	10872.0	45013.0	21.0	1274.600894	930752.0	NaN	1.861445	84000021	USA	20833.037691	NaN
Louisiana	US	2020-09-14 04:30:37	31.1695	-91.8678	157455	5235	140440.0	11780.0	22.0	3387.007469	2066730.0	NaN	3.324759	84000022	USA	44457.336677	NaN
Maine	US	2020-09-14 04:30:37	44.6939	-69.3819	4863	136	4226.0	501.0	23.0	361.773292	330168.0	NaN	2.796628	84000023	USA	24562.197034	NaN
Maryland	US	2020-09-14 04:30:37	39.0639	-76.8021	116110	3838	7225.0	105047.0	24.0	1920.544918	1455845.0	NaN	3.305486	84000024	USA	24080.748568	NaN

NYU Langone Health

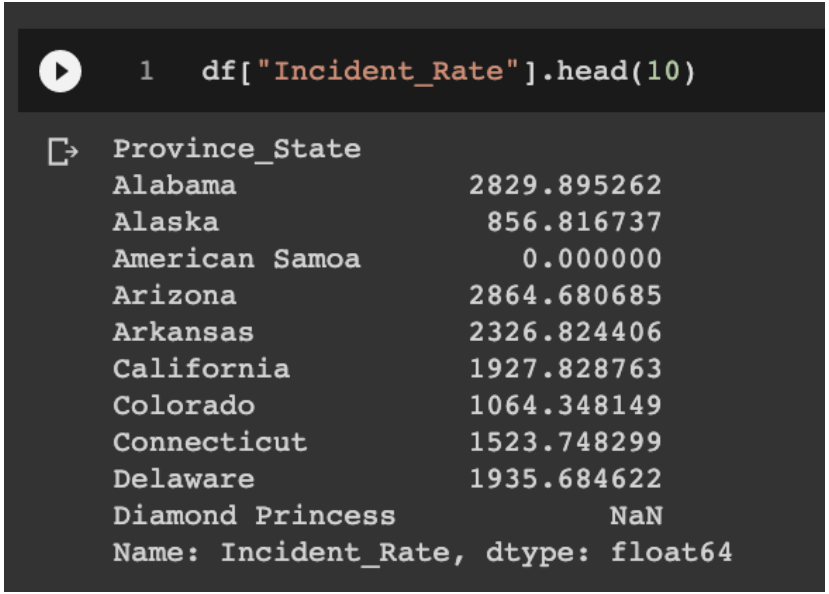
Pandas – Data Structures – Pandas Series

- One-dimensional data structure
- Includes axis labels
  - 'Province_State' in the example
  - Axis labels can be used to pull out individual values

```
df["Incident_Rate"]["Alabama"]

=> 2829.895261957279
```

- Pandas will automatically try to determine the 'dtype' of each series when loading a csv
  - This can be changed using the '`.astype(<type>)`' command
    - <type> will generally be a numpy type (np.int16, np.str, …)
  - This command returns a new series (non-destructive)



NYU Langone Health

Pandas – Data Structures – Series – Setting the type

```
[25]  1  df["FIPS"].head(10)

Province_State
Alabama                1.0
Alaska                 2.0
American Samoa        60.0
Arizona                4.0
Arkansas               5.0
California             6.0
Colorado               8.0
Connecticut            9.0
Delaware              10.0
Diamond Princess    88888.0
Name: FIPS, dtype: float64
```

```
1  df["FIPS"] = df["FIPS"].astype(np.uint16)
2  df["FIPS"].head(10)

Province_State
Alabama                1
Alaska                 2
American Samoa        60
Arizona                4
Arkansas               5
California             6
Colorado               8
Connecticut            9
Delaware              10
Diamond Princess    23352
Name: FIPS, dtype: uint16
```

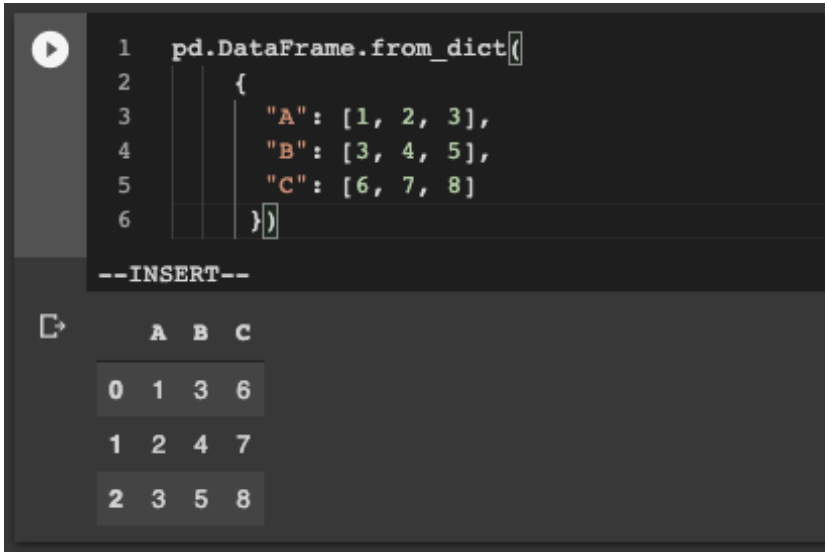

NYU Langone Health

Pandas – Data Structures – Data Frame

- DataFrame is a two-dimensional structure made up of Pandas Series
- The DataFrame is made up of rows and columns
  - Columns are indexed using a unique key (normally a string)
  - Rows are indexed by default using an ‘int’ starting from zero
    - This can be set to a column if the column is unique (‘Province_State’ in my example df)
- `.to_numpy()` will convert the DataFrame / Series into a regular numpy ndarray
  - DataFrame => 2d Array
  - Series => 1d Array
  - This is often useful when plotting using Matplotlib

Pandas – Loading Data into a Data Frame

- Pandas allows for multiple external data sources
  - Hard Drive : '/Users/Uname/Documents/data.csv'
  - URL : 'https://datasource_url/data/somedata.csv'
- Pandas also has data loaders for most common data structures
  - `pd.read_csv(<path>, sep='\t') # sep is optional`
    - This command is highly customizable (link)
  - `pd.read_hdf(<path>)`
    - Hdf5 files allow for multi-indexing
  - `pd.DataFrame.from_dict(<dictionary>)`
    - Convert a dictionary to a DataFrame



NYU Langone Health

Pandas – Accessing Data – Single Column

- Single columns will always be represented as a Series
  - `df["<column_name>"]`
  - `df.loc[:, "<column_name"]`

- Get a list of column names using
  - `df.columns`

```
1  df["Incident_Rate"].head(10)
```

```
Province_State
Alabama             2829.895262
Alaska               856.816737
American Samoa         0.000000
Arizona             2864.680685
Arkansas            2326.824406
California          1927.828763
Colorado            1064.348149
Connecticut         1523.748299
Delaware            1935.684622
Diamond Princess            NaN
Name: Incident_Rate, dtype: float64
```

Pandas – Accessing Data – Single Row

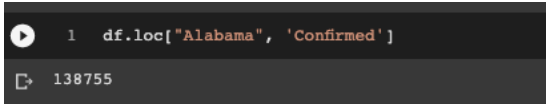
- A row will always be returned as a Series
- `df.iloc[<integer_index>]`
  - Index is the row index starting from 0
- `df.loc["<row_index>"]`

```
1  df.iloc[0] # or df.loc["Alabama"]
--INSERT--
Country_Region                      US
Last_Update             2020-09-14 04:30:37
Lat                           32.3182
Long_                        -86.9023
Confirmed                      138755
Deaths                           2351
Recovered                       54223
Active                          82181
FIPS                                1
Incident_Rate                  2829.9
People_Tested              1.02319e+06
People_Hospitalized                NaN
Mortality_Rate                 1.69435
UID                           84000001
ISO3                               USA
Testing_Rate                  20867.8
Hospitalization_Rate               NaN
Name: Alabama, dtype: object
```


NYU Langone Health

Pandas – Accessing Data – Single Row and Column

- `df.loc[row, column]` can be used to access rows or columns
  - The first index is for rows
  - The second index is for columns
- If both a row and a column index are provided to .loc a value will be returned



NYU Langone Health

Pandas – Accessing Data – Columns

- Multiple columns can be selected using a list of column names
  - This will always return a DataFrame
- `df[["Column1", "Columns2"]]`
- or
- `df.loc[:, ["Column1", "Column2"]]`

```
1  df[["Active", "Recovered"]].head(10)
2  # or df.loc[:, ["Active", "Recovered"]].head(10)
```

Province_State	Active	Recovered
Alabama	82181.0	54223.0
Alaska	3847.0	2377.0
American Samoa	0.0	NaN
Arizona	170373.0	32817.0
Arkansas	6175.0	63063.0
California	747350.0	NaN
Colorado	53149.0	6156.0
Connecticut	40704.0	9142.0
Delaware	8157.0	10077.0
Diamond Princess	49.0	NaN

NYU Langone Health

Pandas – Accessing Data – Rows

- Multiple rows can be selected by using a list of indexes
  - This will always return a DataFrame

```
df.iloc[[idx1, idx2, …]]
```

  - Index is the row index starting from 0

```
df.loc[[“<row_index1>”, “<row_index2>”]]
```

```
1  df.iloc[[0, 2, 3]]
2  # or df.loc[["Alabama", "American Samoa", "Arizona"]]
```

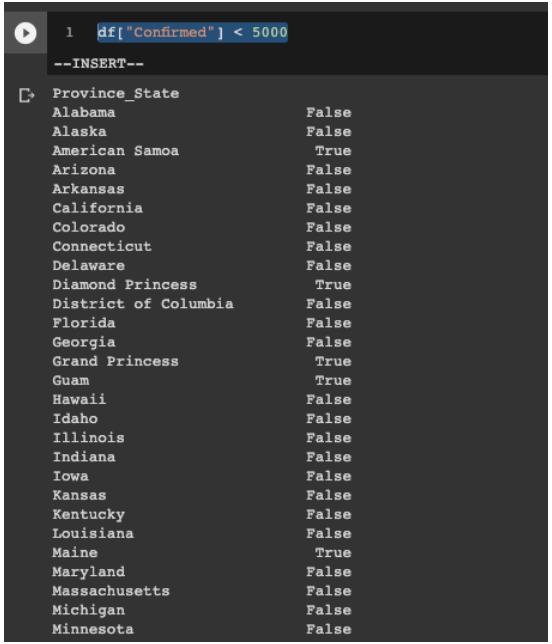
Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	FIPS	Incident_Rate	People_Tested	People_Hospitalized	Mortality_Rate	UID	ISO3	Testing_Rate	Hospitalization_Rate
Alabama	US	2020-09-14 04:30:37	32.3182	-86.9023	138755	2351	54223.0	82181.0	1	2829.895262	1023187.0	NaN	1.694353	84000001	USA	20867.803275	NaN
American Samoa	US	2020-09-14 04:30:37	-14.2710	-170.1320	0	0	NaN	0.0	60	0.000000	1571.0	NaN	NaN	16	ASM	2823.457522	NaN
Arizona	US	2020-09-14 04:30:37	33.7298	-111.4312	208512	5322	32817.0	170373.0	4	2864.680685	1296137.0	NaN	2.552371	84000004	USA	17807.217948	NaN

NYU Langone Health

Pandas – Accessing Data – Subsetting with Expressions

- A list of True / False values can also be used to index the rows of the DataFrame
  - There must be exactly one Boolean per row
  - This lets us use Boolean expressions to subset the df
- Generally the expression is applied to a column

```
df["Confirmed"] < 5000
```



NYU Langone Health

Pandas – Accessing Data – Subsetting with Expressions

- The expression can be used as the .loc parameter
  - A DataFrame will be returned containing the rows where the expression is True

```
[61]  1  df.loc[df["Confirmed"] < 5000]
```

Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	FIPS	Incident_Rate	People_Tested	People_Hospitalized	Mortality_Rate	UID	ISO3	Testing_Rate	Hospitalization_Rate
American Samoa	US	2020-09-14 04:30:37	-14.2710	-170.1320	0	0	NaN	0.0	60	0.000000	1571.0	NaN	NaN	16	ASM	2823.457522	NaN
Diamond Princess	US	2020-09-14 04:30:37	NaN	NaN	49	0	NaN	49.0	88888	NaN	NaN	NaN	0.000000	84088888	USA	NaN	NaN
Grand Princess	US	2020-09-14 04:30:37	NaN	NaN	103	3	NaN	100.0	99999	NaN	NaN	NaN	2.912621	84099999	USA	NaN	NaN
Guam	US	2020-09-14 04:30:37	13.4443	144.7937	1863	23	1118.0	722.0	66	1134.391612	43363.0	NaN	1.234568	316	GUM	26403.984680	NaN
Maine	US	2020-09-14 04:30:37	44.6939	-69.3819	4863	136	4226.0	501.0	23	361.773292	330168.0	NaN	2.796628	84000023	USA	24562.197034	NaN
Northern Mariana Islands	US	2020-09-14 04:30:37	15.0979	145.6739	60	2	29.0	29.0	69	108.806035	14376.0	NaN	3.333333	580	MNP	26069.926012	NaN
Vermont	US	2020-09-14 04:30:37	44.0459	-72.7107	1684	58	1505.0	121.0	50	269.876552	150673.0	NaN	3.444181	84000050	USA	24146.739766	NaN
Virgin Islands	US	2020-09-14 04:30:37	18.3358	-64.8963	1220	19	1144.0	57.0	78	1137.338256	18343.0	NaN	1.557377	850	VIR	17100.160346	NaN
Wyoming	US	2020-09-14 04:30:37	42.7560	-107.3025	4346	42	3768.0	536.0	56	750.917048	86437.0	NaN	0.966406	84000056	USA	14934.886542	NaN

NYU Langone Health

Pandas – Accessing Data – Subsetting with Expressions

- The Boolean expression will return a pandas Series with a True / False values per row
  - Series can be chained using the '&' (and) operator or '|' (or) operator
  - `(df['confirmed'] < 5000) & (df['Active'] > 100)`
    - Each expression must be wrapped in brackets ()
  - Alternatively save each expression result into a variable
    - Combine the variables using the '&' operator or '|' (or) operator

```
1  confirm_thresh = df["Confirmed"] < 5000
2  active_thresh = df["Active"] > 100
3  df.loc[confirm_thresh & active_thresh]
4  # or df.loc[(df["Confirmed"] < 5000) & (df["Active"] > 100)]
```

--INSERT--

Province_State	Country_Region	Last_Update	Lat	Long_	Confirmed	Deaths	Recovered	Active	FIPS	Incident_Rate	People_Tested	People_Hospitalized	Mortality_Rate	UID	ISO3	Testing_Rate	Hospitalization_Rate
Guam	US	2020-09-14 04:30:37	13.4443	144.7937	1863	23	1118.0	722.0	66	1134.391612	43363.0	NaN	1.234568	316	GUM	26403.984680	NaN
Maine	US	2020-09-14 04:30:37	44.6939	-69.3819	4863	136	4226.0	501.0	23	361.773292	330168.0	NaN	2.796628	84000023	USA	24562.197034	NaN
Vermont	US	2020-09-14 04:30:37	44.0459	-72.7107	1684	58	1505.0	121.0	50	269.876552	150673.0	NaN	3.444181	84000050	USA	24146.739766	NaN

NYU Langone Health

Pandas – Summary Statistics

- Pandas has several built-in functions for generating statistics
  - `.mean(), .median(), .min(). max(), .quantile(<quantile ratio>)`
    - DataFrame: Return the statistic for all numeric columns
    - Series: Return the statistic for the series
  - `.describe()`
    - This function will generate a new DataFrame containing multiple statistics for each numeric column
      - count, mean, std, min, 25%, 50%, 75%, max
      - Depending on the column these values might not be useful (FIPS)

Conclusions

- Pandas is a powerful package for data analysis
- The commands covered are sufficient for answering basic questions regarding the data
- In the next lecture we will cover more advanced concepts for organizing data
  - Grouping data
  - Reshaping DataFrames
  - Applying functions across rows / columns

Breakout Session

- The goal for todays session is going to be completing the **Pandas_Basics_Practical_09152020.ipynb** file on Brightspace
- I will complete the notebook to demonstrate
- You will then be divided into random groups using Zoom breakout rooms
- I made a channel on Slack called ‘breakout-discussions’
  - Post any questions you might have into that channel and I will answer them
- After 10-15 minutes we will reconvene
  - At this point we can either
    - start new breakout rooms if this was not enough time
    - start looking at generating plots using Pandas built in functionality (link)

Survey

- https://nyumc.qualtrics.com/jfe/form/SV_6yztFdY5iQpnuBL

Citations

- https://en.wikipedia.org/wiki/Pandas_(software)
- https://numfocus.org/project/pandas