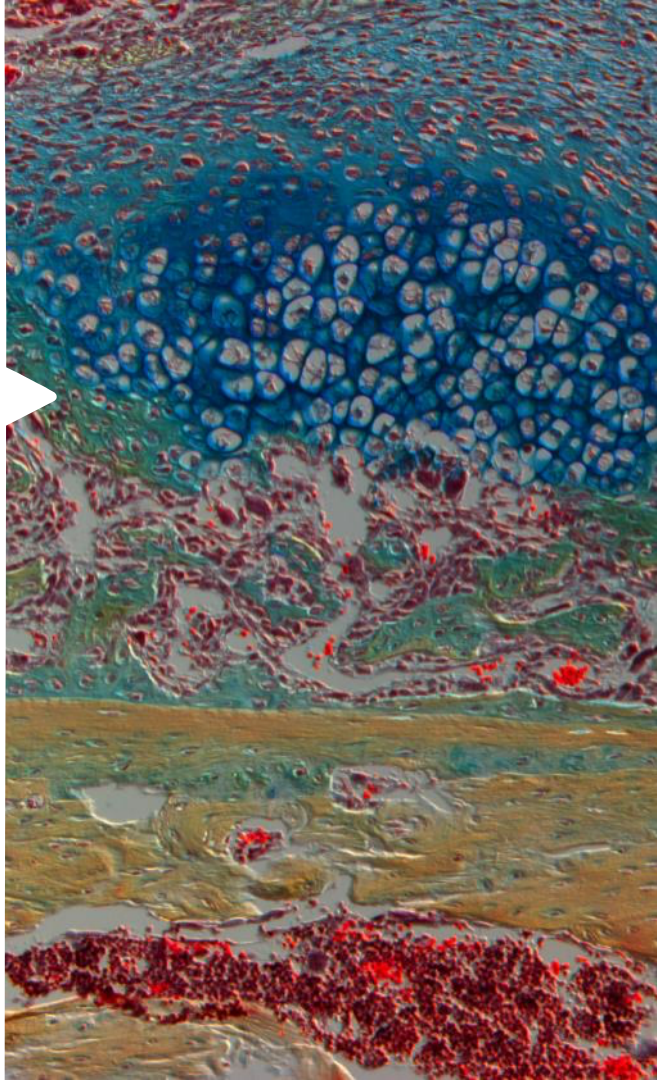
NYU Langone Health

Advanced Pandas and some Seaborn

Mark Grivainis

09 / 16 / 2020



Reading for the Next Lecture

- Golub TR, Slonim DK, Tamayo P, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*. 1999;286(5439):531-537. doi:10.1126/science.286.5439.531
  https://pubmed.ncbi.nlm.nih.gov/10521349/

- Data Portal
  - http://portals.broadinstitute.org/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43

Outline

- Presentation (20-30 min)
  - Advanced Pandas Concepts
- Quiz (5 min to answer, 5 min to recap)
- Break / Questions (10 min)
- Worked Example
  - JHU COVID Timeseries for the US

Pandas - Pivot

Long format data → Wide format data

```
1  df = sns.load_dataset('flights')
2  df
```

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121
...	...	...	...
139	1960	August	606
140	1960	September	508
141	1960	October	461
142	1960	November	390
143	1960	December	432

144 rows × 3 columns

```
1  df.pivot(index='year', columns='month')
```

	passengers											
month	January	February	March	April	May	June	July	August	September	October	November	December
year												
1949	112	118	132	129	121	135	148	148	136	119	104	118
1950	115	126	141	135	125	149	170	170	158	133	114	140
1951	145	150	178	163	172	178	199	199	184	162	146	166
1952	171	180	193	181	183	218	230	242	209	191	172	194
1953	196	196	236	235	229	243	264	272	237	211	180	201
1954	204	188	235	227	234	264	302	293	259	229	203	229
1955	242	233	267	269	270	315	364	347	312	274	237	278
1956	284	277	317	313	318	374	413	405	355	306	271	306
1957	315	301	356	348	355	422	465	467	404	347	305	336
1958	340	318	362	348	363	435	491	505	404	359	310	337
1959	360	342	406	396	420	472	548	559	463	407	362	405
1960	417	391	419	461	472	535	622	606	508	461	390	432

NYU Langone Health

Pandas - Melt

Wide format data → Long format data

```
df = df.droplevel(0, 1).rename(columns=str).reset_index()
df
```

month	year	January	February	March	April	May	June	July	August	September	October	November	December
0	1949	112	118	132	129	121	135	148	148	136	119	104	118
1	1950	115	126	141	135	125	149	170	170	158	133	114	140
2	1951	145	150	178	163	172	178	199	199	184	162	146	166
3	1952	171	180	193	181	183	218	230	242	209	191	172	194
4	1953	196	196	236	235	229	243	264	272	237	211	180	201
5	1954	204	188	235	227	234	264	302	293	259	229	203	229
6	1955	242	233	267	269	270	315	364	347	312	274	237	278
7	1956	284	277	317	313	318	374	413	405	355	306	271	306
8	1957	315	301	356	348	355	422	465	467	404	347	305	336
9	1958	340	318	362	348	363	435	491	505	404	359	310	337
10	1959	360	342	406	396	420	472	548	559	463	407	362	405
11	1960	417	391	419	461	472	535	622	606	508	461	390	432

```
df.melt(id_vars='year')
```

	year	month	value
0	1949	January	112
1	1950	January	115
2	1951	January	145
3	1952	January	171
4	1953	January	196
...	...	...	...
139	1956	December	306
140	1957	December	336
141	1958	December	337
142	1959	December	405
143	1960	December	432

144 rows × 3 columns

NYU Langone Health

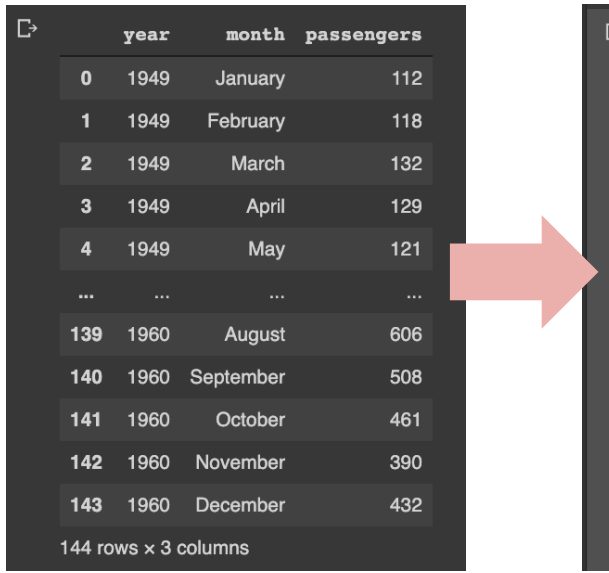
Pandas - Groupby

- Will group rows using the provided column(s)
- On its own it does not do anything
  - It needs an aggregation function
- In the example I used .sum() as the aggregator

```
df[['year', 'passengers']].groupby('year').sum()
```

	year	month	passengers
0	1949	January	112
1	1949	February	118
2	1949	March	132
3	1949	April	129
4	1949	May	121
...	...	...	...
139	1960	August	606
140	1960	September	508
141	1960	October	461
142	1960	November	390
143	1960	December	432

144 rows × 3 columns



year	passengers
1949	1520
1950	1676
1951	2042
1952	2364
1953	2700
1954	2867
1955	3408
1956	3939
1957	4421
1958	4572
1959	5140
1960	5714

NYU Langone Health

Pandas - Groupby

- What is happening in this example?
- Why are all the year results the same?
- What aggregation function did I use?

```
1  df.groupby(['month']).mean()
```

month	year	passengers
January	1954.5	241.750000
February	1954.5	235.000000
March	1954.5	270.166667
April	1954.5	267.083333
May	1954.5	271.833333
June	1954.5	311.666667
July	1954.5	351.333333
August	1954.5	351.083333
September	1954.5	302.416667
October	1954.5	266.583333
November	1954.5	232.833333
December	1954.5	261.833333

Pandas - Apply

- .apply() will apply a function across rows or columns (axis=1 or axis=0)
- The function will return a Series
- In this example I used a lambda function
  - These are unnamed functions
  - They are used for simple operations when it is overkill to write a full function

```
1  df.apply(lambda x: x['year'], axis=1)
--INSERT--
0       1949
1       1949
2       1949
3       1949
4       1949
        ...
139     1960
140     1960
141     1960
142     1960
143     1960
Length: 144, dtype: int64
```

Pandas - Apply

```
def set_year_bin(row):
  year = row["year"]
  start_year = (((year - 1946) // 5) * 5) + 1946
  return "{}-{}".format(start_year, start_year+4)

df.apply(set_year_bin, axis=1)
```

```
--INSERT--
```

```
0       1946-1950
1       1946-1950
2       1946-1950
3       1946-1950
4       1946-1950
          ...
139     1956-1960
140     1956-1960
141     1956-1960
142     1956-1960
143     1956-1960
Length: 144, dtype: object
```

```
def set_year_bin(row):
  year = row["year"]
  start_year = (((year - 1946) // 5) * 5) + 1946
  return "{}-{}".format(start_year, start_year+4)

df['year_bin'] = df.apply(set_year_bin, axis=1)
df
```

	year	month	passengers	year_bin
0	1949	January	112	1946-1950
1	1949	February	118	1946-1950
2	1949	March	132	1946-1950
3	1949	April	129	1946-1950
4	1949	May	121	1946-1950
...	...	...	...	...
139	1960	August	606	1956-1960
140	1960	September	508	1956-1960
141	1960	October	461	1956-1960
142	1960	November	390	1956-1960
143	1960	December	432	1956-1960

144 rows × 4 columns

NYU Langone Health

Pandas - Agg

- Aggregates data
- Like .apply, but normally used with summary statistics functions
- If you pass a list of functions it will return a new DataFrame with a row for each function

```
[114]  1  df.agg(['mean', 'min', 'max'])
```

	year	passengers	year_bin
mean	1954.5	280.298611	NaN
min	1949.0	104.000000	1946-1950
max	1960.0	622.000000	1956-1960

Pandas – Groupby and Agg

```
1  df.groupby('year_bin').agg(['mean', 'min', 'max', 'count'], axis=0)
--NORMAL--
```

	year				passengers			
	mean	min	max	count	mean	min	max	count
year_bin								
1946-1950	1949.5	1949	1950	24	133.166667	104	170	24
1951-1955	1953.0	1951	1955	60	223.016667	145	364	60
1956-1960	1958.0	1956	1960	60	396.433333	271	622	60

Conclusions

- With the functions we have covered in the last two lectures you should be able to manipulate a DataFrame to answer any question you might have
- Sometimes this will require chaining commands together
- The workflow will generally remain the same
  - Load your DataFrame
  - Clean the data (remove meaningless columns, set the types, add any columns that might help)
  - Transform (Create wide / long DataFrames where appropriate, use Groupby's)
  - Visualize
- The only concept we have not covered is merging DataFrames (merge, concat, join)

Survey Link

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

NYU Langone Health

Citations

- https://pandas.pydata.org/docs/
- https://seaborn.pydata.org/api.html