Data Wrangling

Presenter: Steve Baskauf
steve.baskauf@vanderbilt.edu

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

CodeGraf landing page

- vanderbi.lt/codegraf

Data "wrangling"

- a.k.a. data "munging"
- Can involve cleaning, reformatting, summarizing, and changing the data organization to make it more fit for some use like visualization.
- A very large topic – we are only scratching the surface.
- See chapters 5, 7, and 8 in Python for Data Analysis

Basic DataFrame manipulation

Jean & Alexander Heard
LIBRARIES

Column vs. index label

integer index

regular column

	State	Commercial	Electric Power	Residential	Industrial	Transportation	Total
0	Alabama	2.22	55.25	1.87	21.06	34.69	115.09
1	Alaska	2.03	2.75	1.50	16.78	11.85	34.91
2	Arizona	2.87	44.28	2.19	4.59	33.08	87.01
3	Arkansas	2.94	30.22	1.66	8.21	19.38	62.41
4	California	18.87	36.57	24.11	68.84	212.95	361.35

index label

State	Commercial	Electric Power	Residential	Industrial	Transportation	Total
Alabama	2.22	55.25	1.87	21.06	34.69	115.09
Alaska	2.03	2.75	1.50	16.78	11.85	34.91
Arizona	2.87	44.28	2.19	4.59	33.08	87.01
Arkansas	2.94	30.22	1.66	8.21	19.38	62.41
California	18.87	36.57	24.11	68.84	212.95	361.35

Ways to make changes

- Assign to a named view

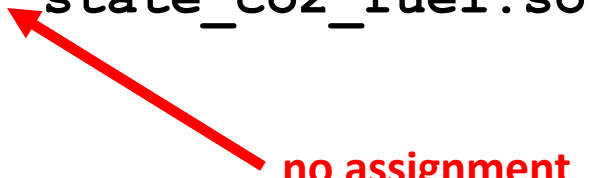
```
sorted_view = state_co2_fuel.sort_values(by='Total mmt')
```

- Assign to a named copy

```
sorted_copy = state_co2_fuel.copy().sort_values(by='Total mmt')
```

- Perform operation "inplace"

```
state_co2_fuel.sort_values(by='Total mmt', inplace=True)
```



Removing rows and columns

Jean & Alexander Heard
LIBRARIES

"Axes" of a data frame

axis 1
column axis

axis names

axis 0
row axis

Sector State	Commercial	Electric Power	Industrial	Residential	Transportation
Alabama	43.996634	1463.430990	503.697916	64.751214	678.018321
Alaska	50.408486	60.772091	406.923918	37.226044	303.227119
Arizona	42.886448	902.591302	96.147374	42.705125	650.674600
Arkansas	41.095011	534.948110	213.471009	52.310787	407.280788
California	329.118829	926.954976	1540.460128	602.205615	4449.921497

Handling missing data

- Pandas has a method for broadly replacing missing data: **`.fillna()`**
- Selection is also possible using **`.isnull()`** and **`.notnull()`** to generate boolean array to be used for selection indexing.

Sorting rows

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

Slicing columns and rows

Jean & Alexander Heard
LIBRARIES

Recall:

- use `.loc[]` for label indices.
- use `.iloc[]` for integer indices.
- only the first index is required to slice rows
- to slice columns, specify the row as `:` , then the column range.
- by default, slices are only views of the data, not copies.

Selecting data

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

How selecting works

- A boolean operation is done on a column. Any common operation (**==**, **<**, **>**, etc.) is possible.
- That generates a series of boolean (**`True`** or **`False`**) the same length as the number of table rows.
- If the series item corresponding to the row is **`True`**, the row is included. If the series item for that row is **`False`**, the row is excluded.
- The resulting DataFrame maintains the indices of the original DataFrame.

Selection indexing process

`organism_info`

index	group (0)	number legs (1)
'lizard' (0)	'reptile'	4
'spider' (1)	'arachnid'	8
'worm' (2)	'annelid'	0
'bee' (3)	'insect'	6

Insert this sequence as the index (in the square brackets).

organism_info['number legs'] > 5
False
True
False
True

index	group (0)	number legs (1)
'spider' (1)	'arachnid'	8
'bee' (3)	'insect'	6

```
organism_info[organism_info['number legs'] > 5]
```

Rearranging data

Jean & Alexander Heard
LIBRARIES

Transposing

- Reverse columns and rows
- `.transpose()`
- `.T`

State	Coal fraction	Petroleum fraction	Natural Gas fraction
Texas	0.191108	0.488192	0.320715
California	0.008330	0.661547	0.330123
Florida	0.174911	0.498957	0.326132
Pennsylvania	0.317005	0.350168	0.332827

State	Texas	California	Florida	Pennsylvania
Coal fraction	0.191108	0.008330	0.174911	0.317005
Petroleum fraction	0.488192	0.661547	0.498957	0.350168
Natural Gas fraction	0.320715	0.330123	0.326132	0.332827

Grouping

- Grouping can be used:
  - to pull out subsets of rows based on values in certain columns.
  - to collapse groups of values by summarizing using **`.sum()`**, **`.mean()`**, etc.

	State	Sector	1990	1991	1992	1993	1994	1995	1996	1997
0	Alabama	Commercial	2.429060	1.999039	2.102710	2.050460	2.054954	1.962636	2.161170	2.417254
1	Alabama	Electric Power	50.279123	54.175721	56.931239	62.817504	58.815941	64.407420	69.577493	67.969353
2	Alabama	Industrial	25.152603	25.356429	28.357535	26.182511	27.122844	27.835798	28.695608	28.711090
3	Alabama	Residential	3.090438	3.015759	3.203561	3.420545	3.301292	3.311796	3.726128	3.342524
4	Alabama	Transportation	28.132113	28.696699	29.385966	29.511222	30.660619	32.150128	31.540431	31.169400
5	Alaska	Commercial	2.197297	2.232937	2.514159	2.560882	2.585992	2.501824	2.710371	2.562721
6	Alaska	Electric Power	2.606027	2.464171	2.293176	2.318983	2.332361	2.409933	2.531790	2.735766
7	Alaska	Industrial	15.828467	17.436042	18.873554	18.505899	18.061311	21.438993	22.526757	21.167328
8	Alaska	Residential	1.580325	1.597610	1.742957	1.731159	1.773315	1.812679	1.795867	1.703034
9	Alaska	Transportation	12.090427	11.168711	10.859193	11.026845	11.193992	12.334794	12.010410	13.488207
10	Arizona	Commercial	1.899725	1.832979	1.776843	1.738700	1.828961	1.792671	1.875969	1.978731
11	Arizona	Electric Power	32.521749	32.757197	35.365378	36.583832	38.013400	32.296326	32.292012	34.998070
12	Arizona	Industrial	3.861071	3.918609	3.925070	3.941926	4.343488	4.677597	4.862293	4.970828

State	1990	1991	1992	1993	1994	1995	1996	1997
Alabama	109.083336	113.243647	119.981011	123.982241	121.955649	129.667778	135.700829	133.609622
Alaska	34.302543	34.899472	36.283039	36.143767	35.946970	40.498223	41.575195	41.657056
Arizona	62.943712	63.758336	66.557229	69.029594	71.694407	66.621808	68.480454	71.606798
Arkansas	51.245592	50.304929	51.934609	51.128270	55.011013	58.353253	60.999313	59.979005
California	362.979853	351.326227	355.506272	345.588203	362.161567	351.282701	352.172619	355.242843
Colorado	65.334483	67.347537	68.324216	72.145227	72.616541	72.680602	75.689976	75.903405

Changing the DataFrame organization

Jean & Alexander Heard
LIBRARIES

Multiple labels for the row index

- More than one column can be set as labels for the row index
- Use a list of column names with **`.set_index()`** instead of a single name:

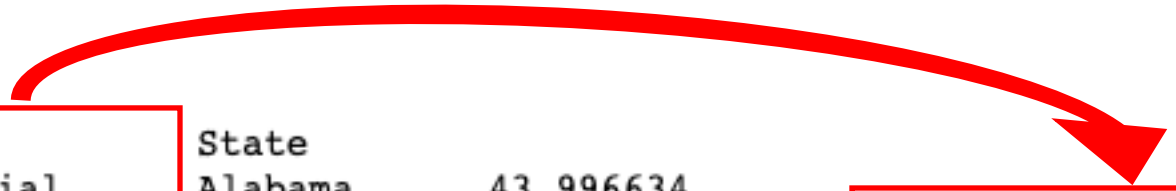
```
state_co2.set_index(['Sector', 'State'])
```

- The row can be identified by a unique combination of labels.

		1990	1991	1992	1993
Sector	**State**				
Commercial	**Alabama**	2.429060	1.999039	2.102710	2.050460
Electric Power	**Alabama**	50.279123	54.175721	56.931239	62.817504
Industrial	**Alabama**	25.152603	25.356429	28.357535	26.182511
Residential	**Alabama**	3.090438	3.015759	3.203561	3.420545
Transportation	**Alabama**	28.132113	28.696699	29.385966	29.511222

Stacked vs. unstacked categories

- **Unstacking** changes a label column from a "grouping variable" to column headers.
- The data items change from a single "stack" (long) to multiple columns. (wide)
- **Stacking** is the reverse process



Sector	State	
Commercial	Alabama	43.996634
Electric Power	Alabama	1463.430990
Industrial	Alabama	503.697916
Residential	Alabama	64.751214
Transportation	Alabama	678.018321
Commercial	Alaska	50.408486
Electric Power	Alaska	60.772091
Industrial	Alaska	406.923918
Residential	Alaska	37.226044
Transportation	Alaska	303.227119
Commercial	Arizona	42.886448
Electric Power	Arizona	902.591302

long

Sector	Commercial	Electric Power	Industrial	Residential	Transportation
State					
Alabama	43.996634	1463.430990	503.697916	64.751214	678.018321
Alaska	50.408486	60.772091	406.923918	37.226044	303.227119
Arizona	42.886448	902.591302	96.147374	42.705125	650.674600
Arkansas	41.095011	534.948110	213.471009	52.310787	407.280788
California	329.118829	926.954976	1540.460128	602.205615	4449.921497

wide

`pd.melt()` function

- Converts **wide** tables into **long** tables
- Column index labels are turned into generic column data.
- Column index name is used as label of the created column, data items are labelled "value".

Sector	State	Commercial	Electric Power	Industrial	Residential	Transportation
0	Alabama	43.996634	1463.430990	503.697916	64.751214	678.018321
1	Alaska	50.408486	60.772091	406.923918	37.226044	303.227119
2	Arizona	42.886448	902.591302	96.147374	42.705125	650.674600
3	Arkansas	41.095011	534.948110	213.471009	52.310787	407.280788
4	California	329.118829	926.954976	1540.460128	602.205615	4449.921497

wide

melt

	State	Sector	value
0	Alabama	Commercial	43.996634
1	Alaska	Commercial	50.408486
2	Arizona	Commercial	42.886448
3	Arkansas	Commercial	41.095011
4	California	Commercial	329.118829

long

`.pivot()` method

- Converts **long** tables into **wide** tables
- Generic column data are turned into **row index labels**.
- Generic column data are turned into **column index labels**.
- Data column data are used to populate the table **cells**.

	State	Sector	value
0	Alabama	Commercial	43.996634
1	Alaska	Commercial	50.408486
2	Arizona	Commercial	42.886448
3	Arkansas	Commercial	41.095011
4	California	Commercial	329.118829

long

pivot

Sector State	Commercial	Electric Power	Industrial	Residential	Transportation
Alabama	43.996634	1463.430990	503.697916	64.751214	678.018321
Alaska	50.408486	60.772091	406.923918	37.226044	303.227119
Arizona	42.886448	902.591302	96.147374	42.705125	650.674600
Arkansas	41.095011	534.948110	213.471009	52.310787	407.280788
California	329.118829	926.954976	1540.460128	602.205615	4449.921497

wide

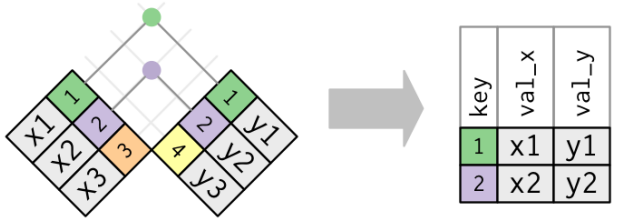
State Sector	Alabama	Alaska	Arizona	Arkansas	California	Colorado	Connecticut	Delaware
Commercial	43.996634	50.408486	42.886448	41.095011	329.118829	86.647082	82.356010	14.317510
Electric Power	1463.430990	60.772091	902.591302	534.948110	926.954976	770.034857	188.943315	125.452611
Industrial	503.697916	406.923918	96.147374	213.471009	1540.460128	212.501061	56.091965	82.912278
Residential	64.751214	37.226044	42.705125	52.310787	602.205615	143.052686	176.706406	23.828466
Transportation	678.018321	303.227119	650.674600	407.280788	4449.921497	535.828002	340.477530	101.667536

Joins

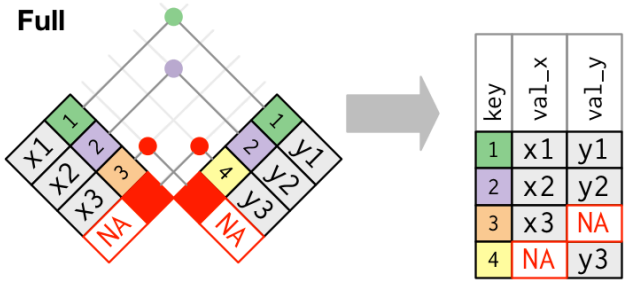
Jean & Alexander Heard
LIBRARIES

Joins

- Joins **merge** data from **multiple tables** (DataFrames)
- **Keys** are the columns used to match table rows
- **Inner join** only outputs rows with matching keys



- **Full outer join** includes rows that don't match (with NaN values inserted)
- Many other permutations



Diagrams from https://r4ds.had.co.nz/relational-data.html

Performing merges (joins) in pandas

- **`pd.merge()`** function specifies both tables, join key(s), and join type.
- The merge can be done using generic column values or index labels.
- See documentation for details on handling overlapping column labels, specifying different keys in the two tables, other join types, etc.

Remote Support for Teaching and Research Needs

- Access to digital collections 24/7
- Skype consultations with your subject librarian
- Ask a Librarian: an easy way to submit a question via email
- Live chat available from the Library home page

Jean & Alexander Heard
LIBRARIES

NEED HELP? ASK A LIBRARIAN!

https://www.library.vanderbilt.edu/ask-librarian.php

Jean & Alexander Heard
LIBRARIES