

Data Wrangling(II): Munging, Tidy Data, and Working with Multiple Data Tables

Nicholas Mattei, *Tulane University*

CMPS3660 – Introduction to Data Science – Fall 2019

<https://rebrand.ly/TUDataScience>



Many Thanks

Slides based off Introduction to Data Science from John P. Dickerson -

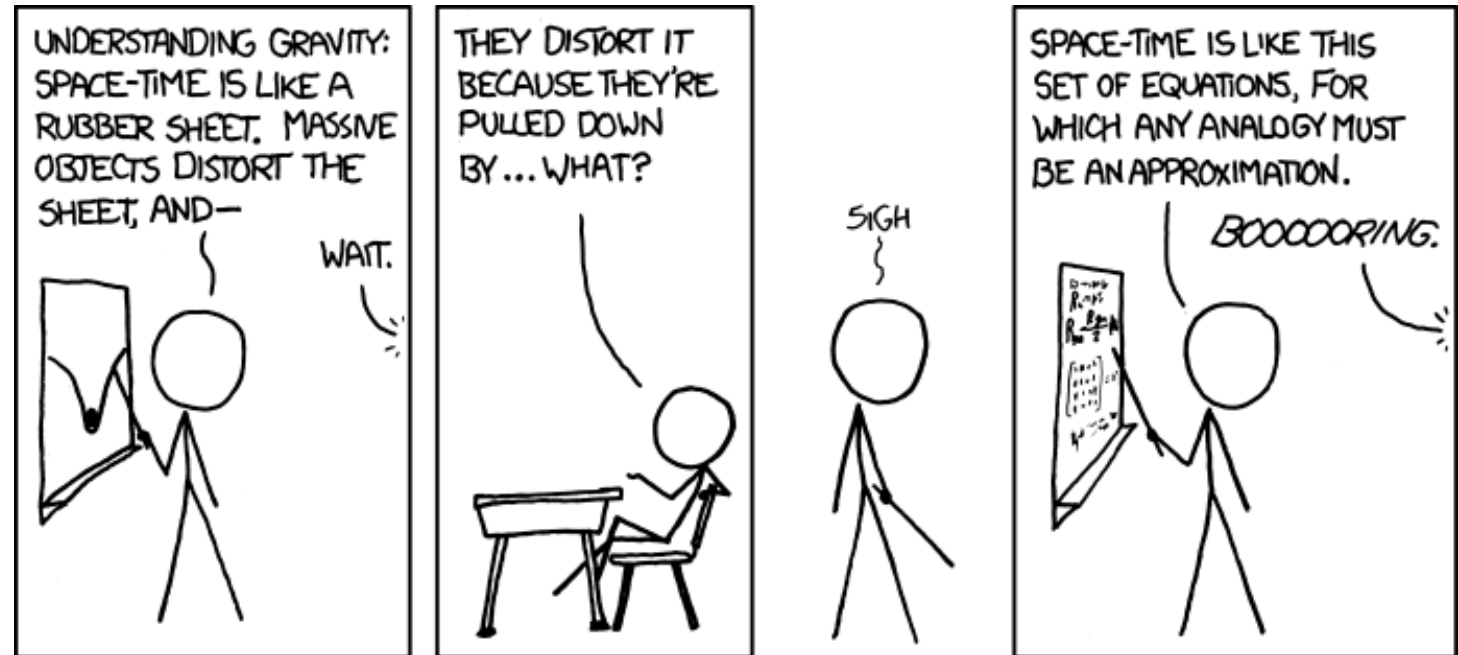
<https://cmsc320.github.io/>

Announcements

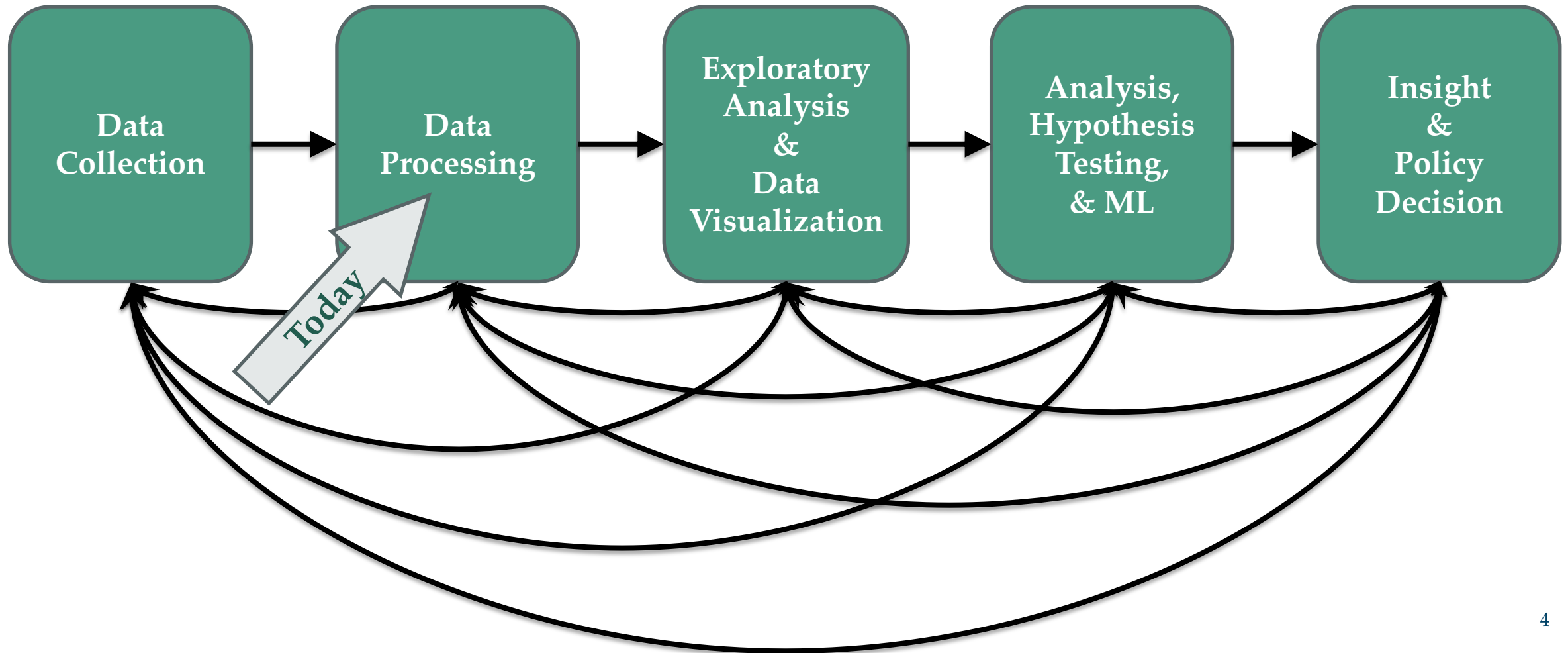
- Project1 and Milestone1 Updates
 - Reading really important here!

Next Couple of Lectures (Till Midterm)

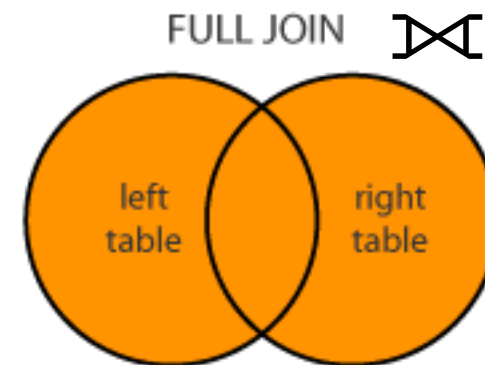
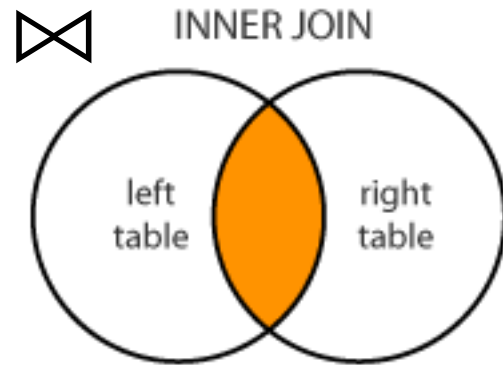
- Tables in the Abstract
 - How, Why
 - Operations
- Principles of Tidy Data
- Tables in Pandas
- Tables in SQL and RMDBS
- 2 More Labs.



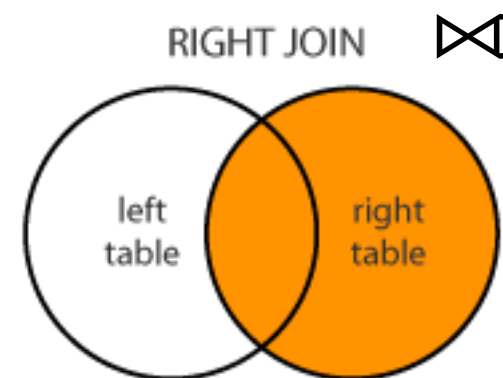
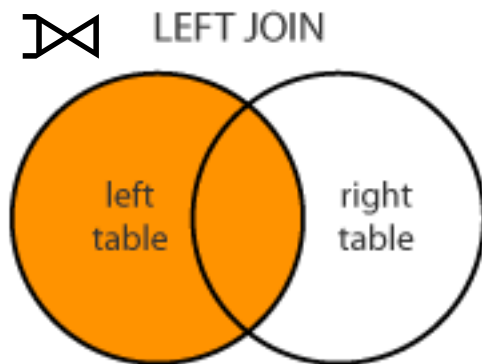
The Data LifeCycle



Types of Joins



In Pandas this is called a
FULL OUTER JOIN!



Quick Review

- Tables: A simple, common abstraction
 - Subsumes a set of “strings” – a common input, or a list of lists, or a list of dicts with the same keys.
- Operations on tables:
 - Select, Map, Aggregate, Reduce, Join/Merge, Union/Concat, Group By
- *These may have different names!* In Pandas it’s a *merge* while in SQL it’s a *join*.
 - Actually, this isn’t quite right -- Pandas has a *join* command that will only join based on the *index!* It also has a *merge* command that allows for more options – see Lab 7!
 - Pandas also uses *merge* as we’ll see in lab while SQL uses Union
- There can be subtle variations in implementation on different data systems. Remember I’m giving you the high level but you need to *read the docs for your software* when you use this stuff!⁶

ID	A	B	C
1	foo	3	6.6
2	baz	2	4.7
3	foo	4	3.1
4	baz	3	8.0
5	bar	1	1.2
6	bar	2	2.5
7	foo	4	2.3
8	foo	3	8.0

Group By 'A'



A = foo

ID	B	C
1	3	6.6
3	4	3.1
7	4	2.3
8	3	8.0

A = baz

ID	B	C
2	2	4.7
4	3	8.0

A = bar

ID	B	C
5	1	1.2
6	2	2.5

How many tuples in the answer?

Note: A GroupBy should partition the whole table!

When does it not?

ID	A	B	C
1	foo	3	6.6
2	baz	2	4.7
3	foo	4	3.1
4	baz	3	8.0
5	bar	1	1.2
6	bar	2	2.5
7	foo	4	2.3
8	foo	3	8.0

Group By 'A', 'B'



A = foo, B=3

ID	C
1	6.6
8	8.0

A = foo, B=4

ID	C
3	3.1
7	2.3

A = baz, B=2

ID	C
2	4.7

A = baz, B=3

ID	C
4	8.0

A = bar, B=1

ID	C
5	1.2

A = bar, B=2

ID	C
6	2.5

How many groups in the answer?

ID	A	B
1	foo	3
2	bar	2
4	foo	4
5	foo	3



ID	C
2	1.2
4	2.5
6	2.3
7	8.0



ID	A	B	C
2	bar	2	1.2
4	foo	4	2.5

How many tuples in the answer?

ID	A	B
1	foo	3
2	bar	2
4	foo	4
5	foo	3



ID	C
2	1.2
4	2.5
6	2.3
7	8.0



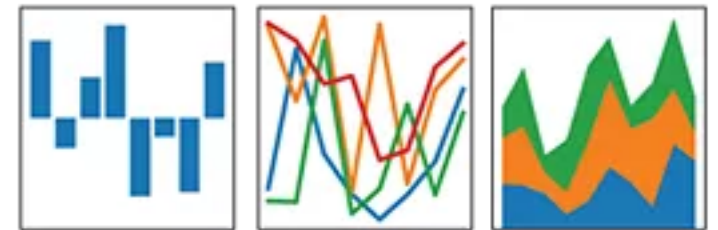
ID	A	B	C
1	foo	3	NaN
2	bar	2	1.2
4	foo	4	2.5
5	foo	3	NaN
6	NaN	NaN	2.3
7	NaN	NaN	8.0

How many tuples in the answer?

Pandas: History

- Written by: Wes McKinney
 - Started in 2008 to get a high-performance, flexible tool to perform quantitative analysis on financial data
- Highly optimized for performance, with critical code paths written in Cython or C
- Key constructs:
 - Series (like a NumPy Array)
 - DataFrame (like a Table or Relation, or R data.frame)
- Foundation for Data Wrangling and Analysis in Python

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$


Pandas: Series

index		values
A	→	5
B	→	6
C	→	12
D	→	-5
E	→	6.7

- Subclass of `numpy.ndarray`
- Data: any type
- Index labels need not be ordered
- Duplicates possible but result in reduced functionality

Pandas: DataFrame

	columns	foo	bar	baz	qux
index					
A	→	0	x	2.7	True
B	→	4	y	6	True
C	→	8	z	10	False
D	→	-12	w	NA	False
E	→	16	a	18	False

- Each column can have a different type
 - Row and Column index
 - Mutable size: insert and delete columns
- Note the use of word “index” for what we called “key”
 - Relational databases use “index” to mean something else
- Non-unique index values allowed
 - May raise an exception for some operations

Hierarchical Indexes

- Sometimes more intuitive organization of the data
- Makes it easier to understand and analyze higher-dimensional data
 - e.g., instead of 3-D array, may only need a 2-D array
 - What is a cube in n-dimensions?

day		Fri	Sat	Sun	Thur
sex	smoker				
Female	No	3.125	2.725	3.329	2.460
	Yes	2.683	2.869	3.500	2.990
Male	No	2.500	3.257	3.115	2.942
	Yes	2.741	2.879	3.521	3.058

```
first second
bar one 0.469112
two -0.282863
baz one -1.509059
two -1.135632
foo one 1.212112
two -0.173215
qux one 0.119209
two -1.044236
dtype: float64
```

Essential Functionality

- Reindexing to change the index associated with a DataFrame
 - Common usage to interpolate, fill in missing values

```
In [84]: obj3 = Series(['blue', 'purple', 'yellow'], index=[0, 2, 4])
```

```
In [85]: obj3.reindex(range(6), method='ffill')
```

```
Out[85]:
```

```
0    blue
1    blue
2  purple
3  purple
4  yellow
5  yellow
```

Essential Functionality

- “drop” to delete entire rows or columns
- Indexing, Selection, Filtering: very similar to NumPy
- Arithmetic Operations
 - Result index union of the two input indexes.
 - Options to do “fill” while doing these operations.

```
In [128]: s1
```

```
Out[128]:
```

```
a    7.3
```

```
c   -2.5
```

```
d    3.4
```

```
e    1.5
```

```
In [129]: s2
```

```
Out[129]:
```

```
a   -2.1
```

```
c    3.6
```

```
e   -1.5
```

```
f    4.0
```

```
g    3.1
```

```
In [130]: s1 + s2
```

```
Out[130]:
```

```
a    5.2
```

```
c    1.1
```

```
d   NaN
```

```
e    0.0
```

```
f   NaN
```

```
g   NaN
```


Function Application and Mapping

```
In [158]: frame = DataFrame(np.random.randn(4, 3), columns=list('bde'),
.....:                      index=['Utah', 'Ohio', 'Texas', 'Oregon'])
```

```
In [159]: frame
Out[159]:
```

	b	d	e
Utah	-0.204708	0.478943	-0.519439
Ohio	-0.555730	1.965781	1.393406
Texas	0.092908	0.281746	0.769023
Oregon	1.246435	1.007189	-1.296221

```
In [160]: np.abs(frame)
Out[160]:
```

	b	d	e
Utah	0.204708	0.478943	0.519439
Ohio	0.555730	1.965781	1.393406
Texas	0.092908	0.281746	0.769023
Oregon	1.246435	1.007189	1.296221

```
In [161]: f = lambda x: x.max() - x.min()
```

```
In [162]: frame.apply(f)
Out[162]:
```

b	1.802165
d	1.684034
e	2.689627

```
In [163]: frame.apply(f, axis=1)
Out[163]:
```

Utah	0.998382
Ohio	2.521511
Texas	0.676115
Oregon	2.542656

Sorting and Ranking

```
In [169]: obj = Series(range(4), index=['d', 'a', 'b', 'c'])
```

```
In [170]: obj.sort_index()
```

```
Out[170]:
```

```
a    1
b    2
c    3
d    0
```

```
In [187]: frame = DataFrame({'b': [4.3, 7, -3, 2], 'a': [0, 1, 0, 1],
.....:                       'c': [-2, 5, 8, -2.5]})
```

```
In [188]: frame
```

```
Out[188]:
```

```
   a    b    c
0  0  4.3 -2.0
1  1  7.0  5.0
2  0 -3.0  8.0
3  1  2.0 -2.5
```

```
In [189]: frame.rank(axis=1)
```

```
Out[189]:
```

```
   a  b  c
0  2  3  1
1  1  3  2
2  2  1  3
3  2  3  1
```

Descriptive and Summary Statistics

Table 5-10. Descriptive and summary statistics

Method	Description
count	Number of non-NA values
describe	Compute set of summary statistics for Series or each DataFrame column
min, max	Compute minimum and maximum values
argmin, argmax	Compute index locations (integers) at which minimum or maximum value obtained, respectively
idxmin, idxmax	Compute index values at which minimum or maximum value obtained, respectively
quantile	Compute sample quantile ranging from 0 to 1
sum	Sum of values
mean	Mean of values
median	Arithmetic median (50% quantile) of values
mad	Mean absolute deviation from mean value
var	Sample variance of values
std	Sample standard deviation of values
skew	Sample skewness (3rd moment) of values
kurt	Sample kurtosis (4th moment) of values
cumsum	Cumulative sum of values
cummin, cummax	Cumulative minimum or maximum of values, respectively
cumprod	Cumulative product of values
diff	Compute 1st arithmetic difference (useful for time series)
pct_change	Compute percent changes

From: *Python for Data Analysis*; Wes McKinney

Creating Dataframes

- Directly from Dict or Series
- From a Comma-Separated File – CSV file
 - `pandas.read_csv()`
 - Can infer headers/column names if present, otherwise may want to reindex
- From an Excel File
 - `pandas.read_excel()`
- From an HTML Table
 - `pandas.read_html()`
- From a Database using SQL (we'll see soon...)
- From Clipboard, URL, Google Analytics, ...

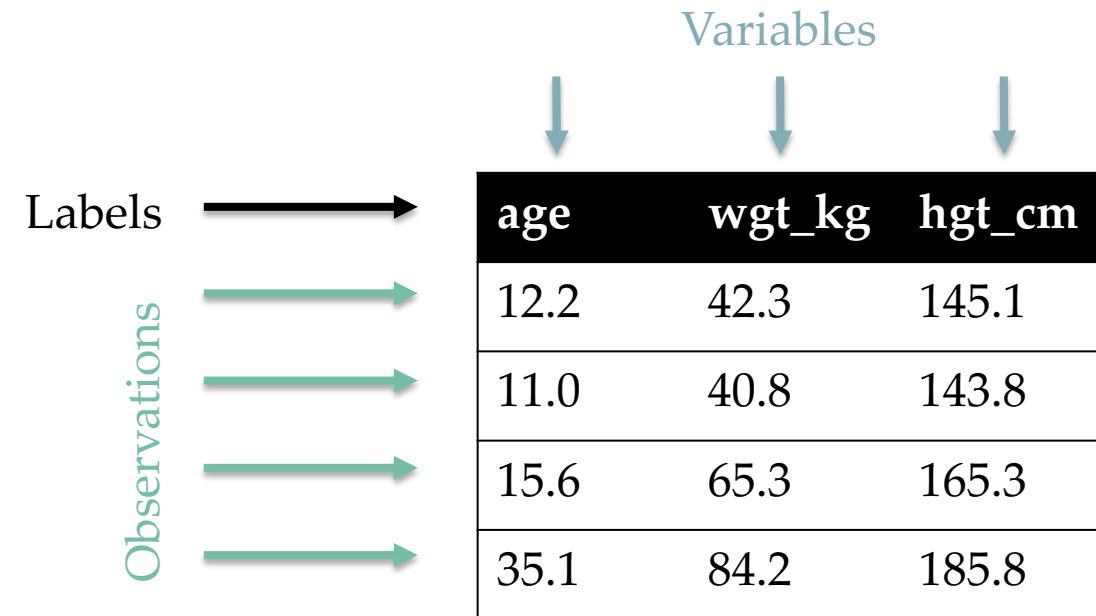
More...

- Unique values, Value counts
- Correlation and Covariance
- Functions for handling missing data – in a few classes
 - `dropna()`, `fillna()`, `isnull()`
- Broadcasting
- Pivoting

- We will (and have) see some of these as we discuss data wrangling, cleaning, etc.

Tidy Data

- The structure Wickham defines as tidy has the following attributes:
 - Each *variable* forms a column and contains *values*
 - Each *observation* forms a row
 - Each type of *observational unit* forms a table
- A few definitions:
 - Variable: A measurement or an attribute. *Height, weight, sex, etc.*
 - Value: The actual measurement or attribute. *152 cm, 80 kg, female, etc.*
 - Observation: All values measure on the same unit. *Each person.*
- But also:
 - Names of files/DataFrames = *description of one dataset*
 - Enforce one data type per dataset (ish)



Labels →

Observations →

Variables

age	wgt_kg	hgt_cm
12.2	42.3	145.1
11.0	40.8	143.8
15.6	65.3	165.3
35.1	84.2	185.8

Example

- Variable: measure or attribute:
 - age, weight, height, sex
- Value: measurement of attribute:
 - 12.2, 42.3kg, 145.1cm, M/F
- Observation: all measurements for an object
 - A specific person is [12.2, 42.3, 145.1, F]

age	wgt_kg	hgt_cm
12.2	42.3	145.1
11.0	40.8	143.8
15.6	65.3	165.3
35.1	84.2	185.8

Tidying Data I

Name	Treatment A	Treatment B
John Smith	-	2
Jane Doe	16	11
Mary Johnson	3	1

???????????????

Name	Treatment A	Treatment B	Treatment C	Treatment D
John Smith	-	2	-	-
Jane Doe	16	11	4	1
Mary Johnson	3	1	-	2

???????????????

Tidying Data II

In a few
lectures ...

Name	Treatment	Result
John Smith	A	-
John Smith	B	2
John Smith	C	-
John Smith	D	-
Jane Doe	A	16
Jane Doe	B	11
Jane Doe	C	4
Jane Doe	D	1
Mary Johnson	A	3
Mary Johnson	B	1
Mary Johnson	C	-
Mary Johnson	D	2

Melting Data I



religion	<\$10k	\$10-20k	\$20-30k	\$30-40k	\$40-50k	\$50-75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Dont know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovahs Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

??????????????

Melting Data II

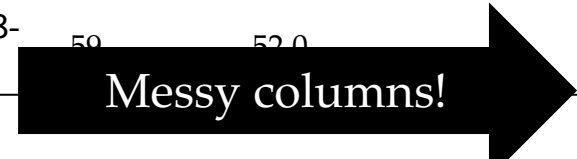
```
f_df = pd.melt(df,  
              ["religion"],  
              var_name="income",  
              value_name="freq")  
f_df = f_df.sort_values(by=["religion"])  
f_df.head(10)
```

religion	income	freq
Agnostic	<\$10k	27
Agnostic	\$30-40k	81
Agnostic	\$40-50k	76
Agnostic	\$50-75k	137
Agnostic	\$10-20k	34
Agnostic	\$20-30k	60
Atheist	\$40-50k	35
Atheist	\$20-30k	37
Atheist	\$10-20k	27
Atheist	\$30-40k	52

More Complicated Example

- Billboard Top 100 data for songs, position on the Top 100 for 75 weeks, with two “messy” bits:
 - Column headers for each of the 75 weeks
 - If a song didn’t last 75 weeks, those columns have are null

year	artist.inverted	track	time	genre	date.entered	date.peak	x1st.week	x2nd.week	...
2000	Destiny's Child	Independent Women Part I	3:38	Rock	2000-09-23	2000-11-18	78	63.0	...
2000	Santana	Maria, Maria	4:18	Rock	2000-02-12	2000-04-08	15	8.0	...
2000	Savage Garden	I Knew I Loved You	4:07	Rock	1999-10-23	2000-01-29	71	48.0	...
2000	Madonna	Music	3:45	Rock	2000-08-12	2000-09-16	41	23.0	...
2000	Aguilera, Christina	Come On Over Baby	3:38	Rock	2000-08-05	2000-10-14	57	47.0	...
2000	Janet	Doesn't Really Matter	4:17	Rock	2000-06-17	2000-08-26	59	52.0	...



More Complicated Example

```
# Keep identifier variables
id_vars = ["year",
           "artist.inverted",
           "track",
           "time",
           "genre",
           "date.entered",
           "date.peaked"]

# Melt the rest into week and rank columns
df = pd.melt(frame=df,
             id_vars=id_vars,
             var_name="week",
             value_name="rank")
```

- Creates one row per week, per record, with its rank

More Complicated Example

```
# Formatting
df["week"] = df['week'].str.extract('(\d+)',
                                     expand=False).astype(int)
df["rank"] = df["rank"].astype(float)
```

```
[..., "x2nd.week", 63.0] → [..., 2, 63]
```

```
# Cleaning out unnecessary rows
df = df.dropna()

# Create "date" columns
df['date'] = pd.to_datetime(
    df['date.entered'] +
    pd.to_timedelta(df['week'], unit='w') -
    pd.DateOffset(weeks=1)
```

More Complicated Example

```
# Ignore now-redundant, messy columns
df = df[["year",
        "artist.inverted",
        "track",
        "time",
        "genre",
        "week",
        "rank",
        "date"]]

df = df.sort_values(ascending=True,
                  by=["year", "artist.inverted", "track", "week", "rank"])

# Keep tidy dataset for future usage
billboard = df

df.head(10)
```

More Complicated Example

year	artist.in verted	track	time	genre	week	rank	date
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	1	87	2000-02-26
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	2	82	2000-03-04
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	3	72	2000-03-11
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	4	77	2000-03-18
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	5	87	2000-03-25
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	6	94	2000-04-01
2000	2 Pac	Baby Don't Cry (Keep Ya Head Up II)	4:22	Rap	7	99	2000-04-08
2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	1	91	2000-09-02
2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	2	87	2000-09-09
2000	2Ge+her	The Hardest Part Of Breaking Up (Is Getting Ba...	3:15	R&B	3	92	2000-09-16

??????????????

More To Do?

- Column headers are values, not variable names?
 - Good to go!
- Multiple variables are stored in one column?
 - Maybe (depends on if genre text in raw data was multiple)
- Variables are stored in both rows and columns?
 - Good to go!
- Multiple types of observational units in the same table?
 - Good to go! One row per song's week on the Top 100.
- A single observational unit is stored in multiple tables?
 - Don't do this!
- Repetition of data?
 - Lots! Artist and song title's text names. Which leads us to ...



SQL And Relational Data

- Relational data:
 - What is a relation, and how do they interact?
- Querying databases:
 - SQL
 - SQLite
 - How does this relate to pandas?
- Joins

