Data Cleaning

+ Project group formation

IMT 547 - Social Media Data Mining and Analysis

2-Feb-2021 (Week 5, Day 9)

Today’s Topics

- Project group formation
  - Project pitches - due Friday midnight
- Left-over EDA lab
- Data cleaning
  - Lab - data cleaning

Project Group Formation

Up until 8:55

Project Group Formation

Project Pitch

Left-over EDA lab

Data Cleaning

Recall the Data Science Workflow

1. Define problem
2. Collect data
3. **Process data <- Data cleaning**
4. Visualize data
5. Analyze data
6. Report

Data carpentry

WRITTEN BY DAVID MIMNO

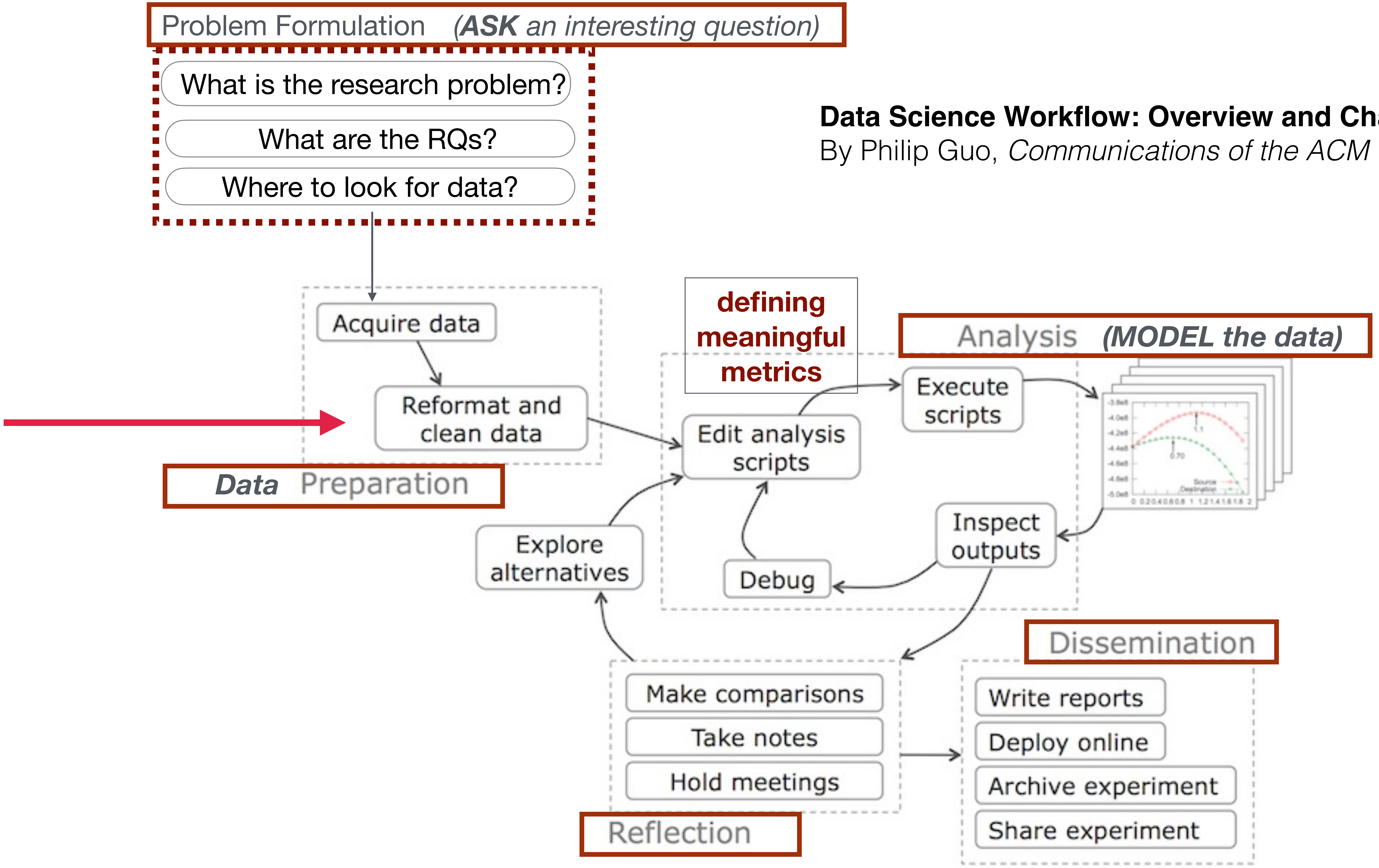
The New York Times has an article titled For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights. Mostly I really like it. The fact that raw data is rarely usable for analysis without significant work is a point I try hard to make with my students. I told them "do not underestimate the difficulty of data preparation". When they turned in their projects, many of them reported that they had underestimated the difficulty of data preparation. Recognizing this as a hard problem is great.

What I'm less thrilled about is calling this "janitor work". For one thing, it's not particularly respectful of custodians, whose work I really appreciate. But it also mischaracterizes what this type of work is about. I'd like to propose a different analogy that I think fits a lot better: *data carpentry*.

Note: data carpentry seems to already be a thing

Data Science Workflow

Data Science Workflow: Overview and Challenges
By Philip Guo, *Communications of the ACM*



Cleaning data values and types

1.Missing data

2.Invalid data (e.g. “Age” = -22)

3.Extreme data (e.g. “Age” = 150)

4.Messy categories (e.g.: major name entry: “Stats”, “Statistics”, “STAT”)

5.Wrong data types (e.g.: integer as string “47”)

8.Duplicates

1. set to NaN (nan, NA, NaN all equivalent)

2. Invalid data - set to NaN

3.Extreme data - set to NaN

4. Messy categories - standardize, e.g. STAT

5.Wrong data types - convert, e.g. int(“47”)

8. Duplicates - eliminate

Working with missing data

1. Find the number of missing values in your data

```
ebola = pd.read_csv('../data/country_timeseries.csv')
```

```
import numpy as np

print(np.count_nonzero(ebola.isnull()))
```

count the total number of missing values in your data

```
1214
```

```
print(np.count_nonzero(ebola['Cases_Guinea'].isnull()))
```

count the total number of missing values for a particular column

```
29
```

Working with missing data

2. Compute With Missing Data

Calculations with missing values will typically return a missing value, unless the function or method called has a means to ignore missing values in its calculations.

```
# skipping missing values is True by default
print(ebola.Cases_Guinea.sum(skipna = True))
```

```
84729.0
```

```
print(ebola.Cases_Guinea.sum(skipna = False))
```

```
nan
```

Working with missing data

3. Remove rows with missing values

drop observations or variables with missing data

Caveat: Depending on how much data is missing, keeping only complete case data can leave you with a useless data set or biased data

```
ebola_dropna = ebola.dropna()
print(ebola_dropna.shape)
```

```
(1, 18)
```

```
print(ebola_dropna)
```

```
          Date  Day  Cases_Guinea  Cases_Liberia  Cases_SierraLeone  \
19  11/18/2014  241        2047.0         7082.0             6190.0

    Cases_Nigeria  Cases_Senegal  Cases_UnitedStates  Cases_Spain  \
19           20.0            1.0                 4.0          1.0

    Cases_Mali  Deaths_Guinea  Deaths_Liberia  Deaths_SierraLeone  \
19         6.0         1214.0          2963.0              1267.0

    Deaths_Nigeria  Deaths_Senegal  Deaths_UnitedStates  \
19             8.0             0.0                  1.0

    Deaths_Spain  Deaths_Mali
```

Working with missing data

4. Imputation

Replacing missing data with substituted values. E.g.: recoding missing values as a 0.

```
print(ebola.fillna(0).iloc[0:10, 0:5])
```

Working with missing data

4. Imputation

Replacing missing data with substituted values: https://en.wikipedia.org/wiki/Imputation_(statistics)

- Fixed value, 0
- Reduce operator on the column: mean, median
  - E.g. mean of data in the same category
- Regression with some other column
  - E.g. PercentCollegeGrad -> IncomePerCapita
- Sample from the column distribution
  - E.g. value frequencies, simulated data imputation.

Tidy Data

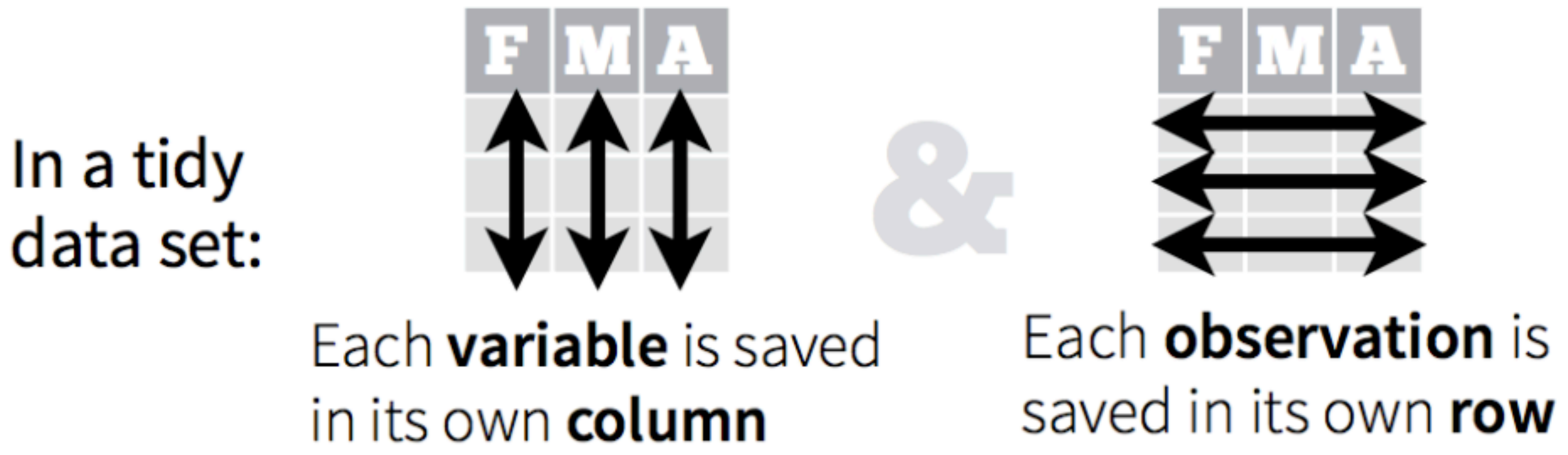
Tidying: Structuring your dataset to facilitate analysis.

Tidy Data

Tidying: Structuring your dataset to facilitate analysis.

What is tidy data?

Tidy data convention: put variables in the columns and observations in the rows.

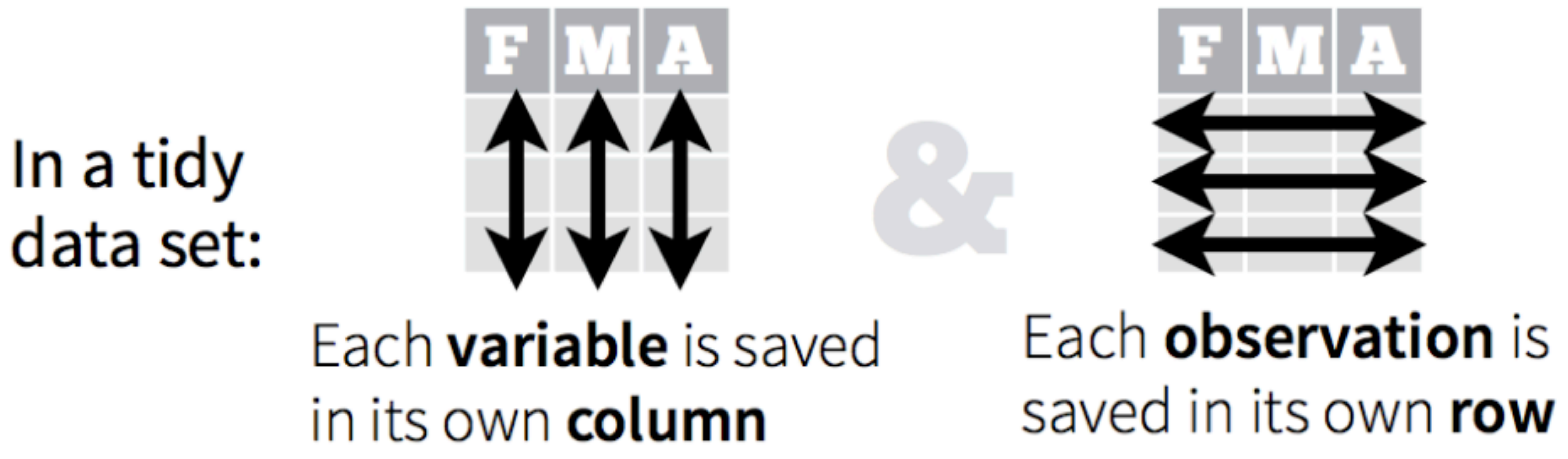


Tidy Data

Tidying: Structuring your dataset to facilitate analysis.

What is tidy data?

Tidy data convention: put variables in the columns and observations in the rows.



Messy

patient	treatment_a	treatment_b
John Smith	n/a	2
Jane Doe	16	11
Mary Johnson	3	1

patient	treatment	result
John Smith	a	n/a
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Clean

Tidy Data

In tidy data:

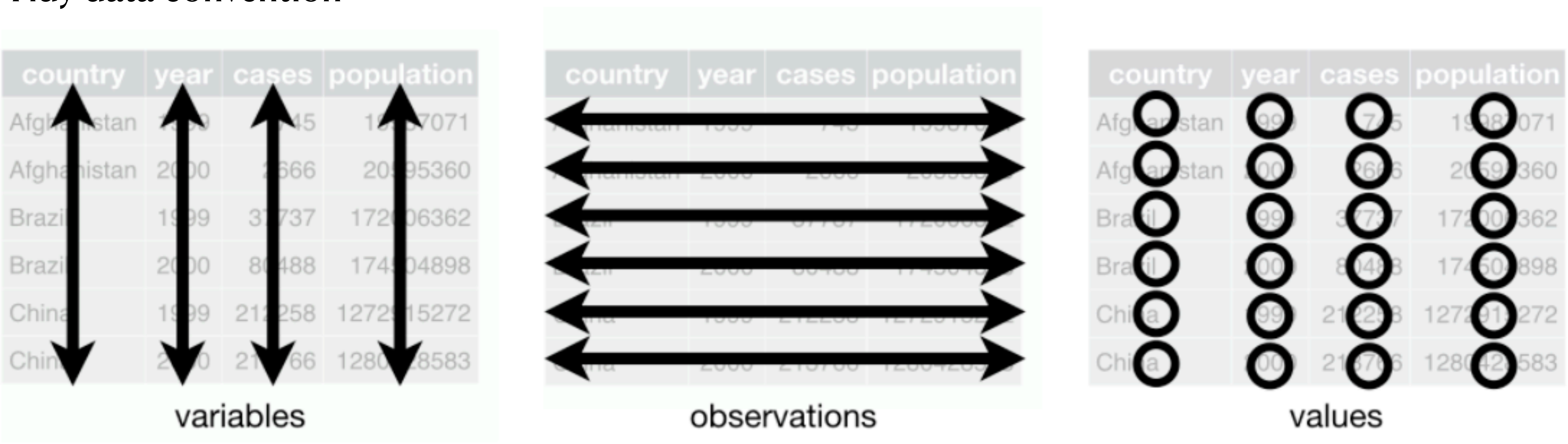
- Each variable must have its own column.
- Each observation must have its own row.
- Each value must have its own cell.

Raw data about countries and their health-index by year

country	year	pop	continent	lifeExp	gdpPercap
Afghanistan	1952	8425333	Asia	28.801	779.4453145
Afghanistan	1957	9240934	Asia	30.332	820.8530296
Afghanistan	1962	10267083	Asia	31.997	853.10071
Afghanistan	1967	11537966	Asia	34.02	836.1971382
Afghanistan	1972	13079460	Asia	36.088	739.9811058
Afghanistan	1977	14880372	Asia	38.438	786.11336
Afghanistan	1982	12881816	Asia	39.854	978.0114388
Afghanistan	1987	13867957	Asia	40.822	852.3959448
Afghanistan	1992	16317921	Asia	41.674	649.3413952
Afghanistan	1997	22227415	Asia	41.763	635.341351
Afghanistan	2002	25268405	Asia	42.129	726.7340548

RAW DATA here: https://raw.githubusercontent.com/OHI-Science/data-science-training/master/data/gapminder.csv

Tidy data convention



Is this data in tidy data form?

country	1999	2000
Afganistan	745	2666
Brazil	37737	80488
China	22258	213766

Common problems with data

A common problem is a dataset where some of the column names are not names of variables, but values of a variable.

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afganistan	745	2666
Brazil	37737	80488
China	22258	213766

Common problems with data

A common problem is a dataset where some of the column names are not names of variables, but values of a variable.

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afganistan	745	2666
Brazil	37737	80488
China	22258	213766

#1: Column headers are values, not variable names

Common problems with data

A common problem is a dataset where some of the column names are not names of variables, but values of a variable.

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afganistan	745	2666
Brazil	37737	80488
China	22258	213766

#1: Column headers are values, not variable names

Is this data in tidy data form?

A common problem is a dataset where some of the column names are not names of variables, but values of a variable.

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

Pivoting table 4 into a longer, tidy form

https://bookdown.org/hneth/ds4psy/7-2-tidy-essentials.html#tidy:longer

Common problems with data

#1: Column headers are values, not variable names (another example)

3 variables ***before***:
Religion (rows), Income (columns), Frequency (cells)

Melt operation to convert columns into rows

3 variables ***after***, 1 per column:
Religion, Income, Frequency

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
Don't 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
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

Table 4: The first ten rows of data on income and religion from the Pew Forum. Three columns, `$75-100k`, `$100-150k` and `>150k`, have been omitted

religion	income	freq
Agnostic	<$10k	27
Agnostic	$10-20k	34
Agnostic	$20-30k	60
Agnostic	$30-40k	81
Agnostic	$40-50k	76
Agnostic	$50-75k	137
Agnostic	$75-100k	122
Agnostic	$100-150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

Common problems with data

#2: Multiple variables stored in one column

country	year	m014	m1524	m2534	m3544	m4554	m5564	m65	mu	f014
AD	2000	0	0	1	0	0	0	0	—	—
AE	2000	2	4	4	6	5	12	10	—	3
AF	2000	52	228	183	149	129	94	80	—	93
AG	2000	0	0	0	0	0	0	1	—	1
AL	2000	2	19	21	14	24	19	16	—	3
AM	2000	2	152	130	131	63	26	21	—	1
AN	2000	0	0	1	2	0	0	0	—	0
AO	2000	186	999	1003	912	482	312	194	—	247
AR	2000	97	278	594	402	419	368	330	—	121
AS	2000	—	—	—	—	1	1	—	—	—

country	year	column	cases
AD	2000	m014	0
AD	2000	m1524	0
AD	2000	m2534	1
AD	2000	m3544	0
AD	2000	m4554	0
AD	2000	m5564	0
AD	2000	m65	0
AE	2000	m014	2
AE	2000	m1524	4
AE	2000	m2534	4
AE	2000	m3544	6
AE	2000	m4554	5
AE	2000	m5564	12
AE	2000	m65	10
AE	2000	f014	3

(a) Molten data

country	year	sex	age	cases
AD	2000	m	0-14	0
AD	2000	m	15-24	0
AD	2000	m	25-34	1
AD	2000	m	35-44	0
AD	2000	m	45-54	0
AD	2000	m	55-64	0
AD	2000	m	65+	0
AE	2000	m	0-14	2
AE	2000	m	15-24	4
AE	2000	m	25-34	4
AE	2000	m	35-44	6
AE	2000	m	45-54	5
AE	2000	m	55-64	12
AE	2000	m	65+	10
AE	2000	f	0-14	3

(b) Tidy data

Common problems with data

#3: Variables stored in both rows and columns

id	year	month	element	d1	d2	d3	d4	d5	d6	d7	d8
MX17004	2010	1	tmax	—	—	—	—	—	—	—	—
MX17004	2010	1	tmin	—	—	—	—	—	—	—	—
MX17004	2010	2	tmax	—	27.3	24.1	—	—	—	—	—
MX17004	2010	2	tmin	—	14.4	14.4	—	—	—	—	—
MX17004	2010	3	tmax	—	—	—	—	32.1	—	—	—
MX17004	2010	3	tmin	—	—	—	—	14.2	—	—	—
MX17004	2010	4	tmax	—	—	—	—	—	—	—	—
MX17004	2010	4	tmin	—	—	—	—	—	—	—	—
MX17004	2010	5	tmax	—	—	—	—	—	—	—	—
MX17004	2010	5	tmin	—	—	—	—	—	—	—	—

id	date	element	value
MX17004	2010-01-30	tmax	27.8
MX17004	2010-01-30	tmin	14.5
MX17004	2010-02-02	tmax	27.3
MX17004	2010-02-02	tmin	14.4
MX17004	2010-02-03	tmax	24.1
MX17004	2010-02-03	tmin	14.4
MX17004	2010-02-11	tmax	29.7
MX17004	2010-02-11	tmin	13.4
MX17004	2010-02-23	tmax	29.9
MX17004	2010-02-23	tmin	10.7

(a) Molten data

id	date	tmax	tmin
MX17004	2010-01-30	27.8	14.5
MX17004	2010-02-02	27.3	14.4
MX17004	2010-02-03	24.1	14.4
MX17004	2010-02-11	29.7	13.4
MX17004	2010-02-23	29.9	10.7
MX17004	2010-03-05	32.1	14.2
MX17004	2010-03-10	34.5	16.8
MX17004	2010-03-16	31.1	17.6
MX17004	2010-04-27	36.3	16.7
MX17004	2010-05-27	33.2	18.2

(b) Tidy data

Reshaping data with pandas

http://pandas.pydata.org/pandas-docs/stable/user_guide/reshaping.html

Melt: to reshape to a long data shape so we can perform

(one or more columns are *identifier variables*, while all other columns, considered *measured variables*)

Melt

```
df3
```

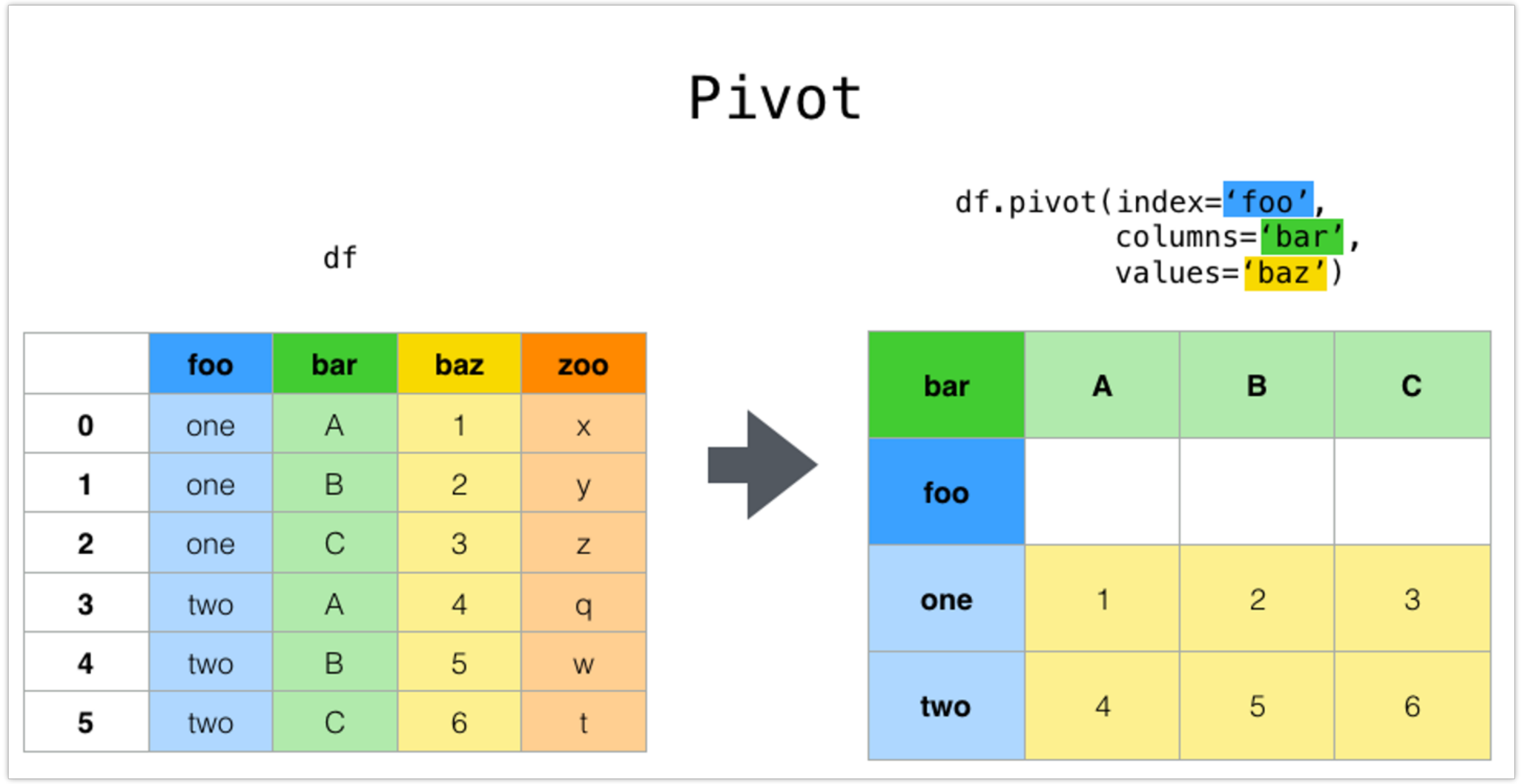
	first	last	height	weight
0	John	Doe	5.5	130
1	Mary	Bo	6.0	150

```
df3.melt(id_vars=['first', 'last'])
```

	first	last	variable	value
0	John	Doe	height	5.5
1	Mary	Bo	height	6.0
2	John	Doe	weight	130
3	Mary	Bo	weight	150

Reshaping data with pandas

Pivot: Reshape data (produce a “pivot” table) based on column values.




```
pivot_table_df=pd.pivot_table(selective_df,index=['Region','Segment'])
```

```
pivot_table_df
```

Region	Segment	Discount	Profit	Quantity	Sales
Central	Consumer	0.252030	7.066046	3.728548	207.946728
	Corporate	0.239822	27.791831	3.869242	234.763466
	Home Office	0.208858	28.398202	3.783105	208.248046
East	Consumer	0.147447	28.040153	3.639891	238.875539
	Corporate	0.144356	26.935666	3.828962	228.516929
	Home Office	0.141036	53.205611	3.810757	253.911805
South	Consumer	0.142124	32.116435	3.792363	233.390180
	Corporate	0.157745	29.833771	3.952941	238.992025
	Home Office	0.143382	16.987626	3.731618	272.996329
West	Consumer	0.107506	34.360409	3.873804	217.033955
	Corporate	0.113958	35.872323	3.781250	235.265911
	Home Office	0.106918	28.949939	3.781086	239.442692

Use case: we have to prepare report across all Regions and Segments **aggregating** the Sales, Discount, Profit and Quantity for each.

```
selective_df=pd.DataFrame(full_data_df, columns= ['Order ID','Order Date','Product ID','Ship Mode','Segment','Country','State','Region','Category',\
                                                   'Sub-Category', 'Sales','Quantity','Discount','Profit'])
```

```
selective_df.head(5)
```

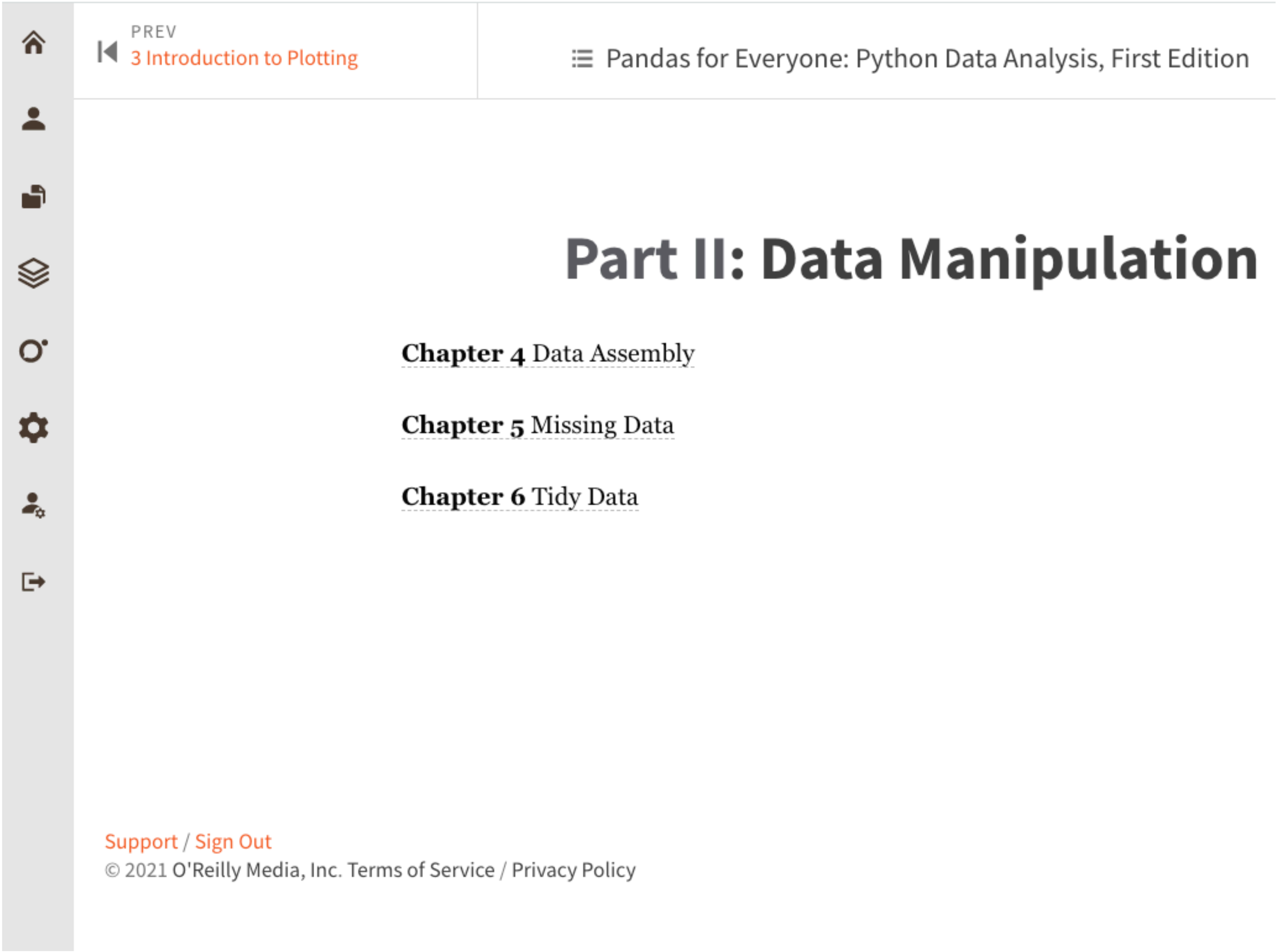
	Order ID	Order Date	Product ID	Ship Mode	Segment	Country	State	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
0	CA-2017-152156	2017-11-08	FUR-BO-10001798	Second Class	Consumer	United States	Kentucky	South	Furniture	Bookcases	261.9600	2	0.00	41.9136
1	CA-2017-152156	2017-11-08	FUR-CH-10000454	Second Class	Consumer	United States	Kentucky	South	Furniture	Chairs	731.9400	3	0.00	219.5820
2	CA-2017-138688	2017-06-12	OFF-LA-10000240	Second Class	Corporate	United States	California	West	Office Supplies	Labels	14.6200	2	0.00	6.8714
3	US-2016-108966	2016-10-11	FUR-TA-10000577	Standard Class	Consumer	United States	Florida	South	Furniture	Tables	957.5775	5	0.45	-383.0310
4	US-2016-108966	2016-10-11	OFF-ST-10000760	Standard Class	Consumer	United States	Florida	South	Office Supplies	Storage	22.3680	2	0.20	2.5164

https://towardsdatascience.com/reshaping-data-with-pandas-19156e8b7af3

Reading Resources

Online book available with UW library access

- Pandas for Everyone, Chapter 5 -- Missing data
- Pandas for Everyone, Chapter 6 -- Tidy data



BREAK

BE back at 9:40am

Lab