Advanced Data Management (CSCI 490/680)

Data Integration

Dr. David Koop

Three Ways to Present the Same Data

	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

Initial Data

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Tidy Data

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

Transpose

[H. Wickham, 2014]

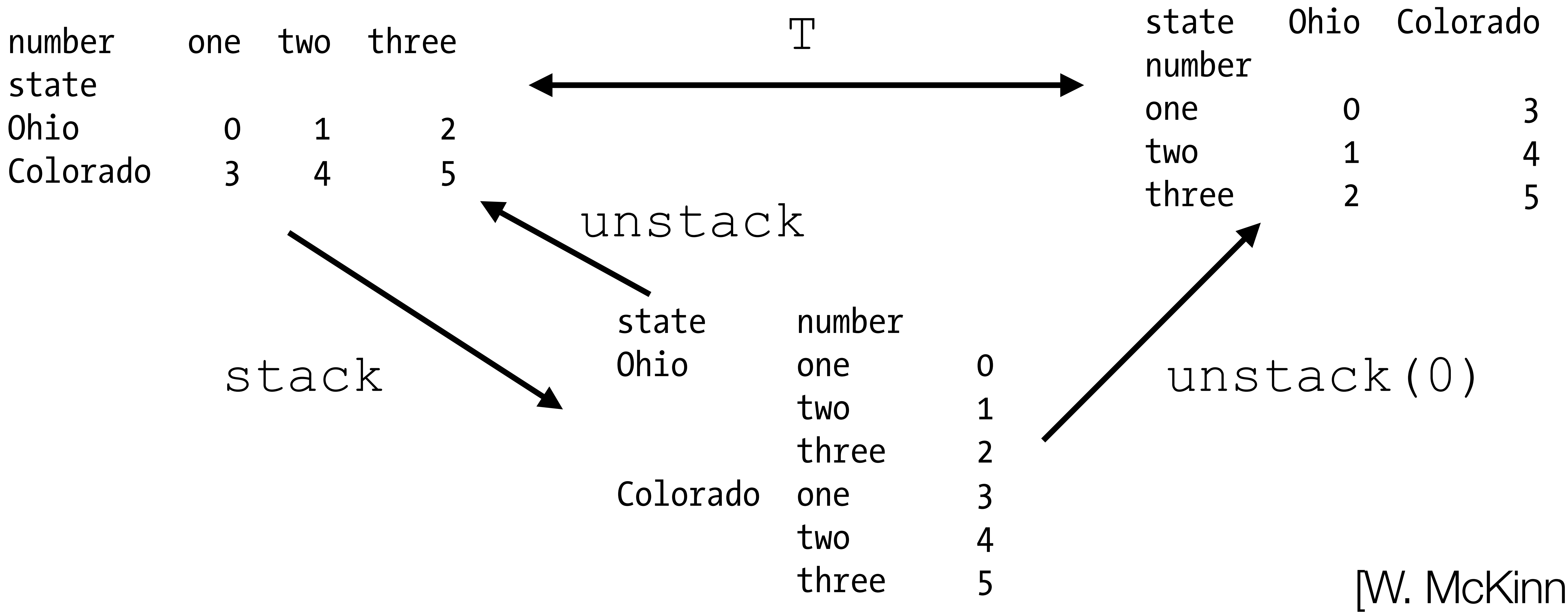
Tidy Data Principles

- **Tidy Data**: Codd's 3rd Normal Form (Databases)
  1. Each variable forms a column
  2. Each observation forms a row
  3. Each type of observational unit forms a table (DataFrame)
- Other structures are **messy data**
- Benefits:
  - Easy for analyst to extract variables
  - Works well for vectorized programming
- Organize variables by their role
  - Fixed variables: describe experimental design, known in advance
  - Measured variables: what is measured in study

[H. Wickham, 2014]

Stack and Unstack

- `stack`: pivots from the columns into rows (may produce a Series!)
- `unstack`: pivots from rows into columns
- unstacking may add missing data
- stacking filters out missing data (unless `dropna=False`)
- can unstack at a different level by passing it (e.g. 0), defaults to innermost level



```
number    one  two  three
state
Ohio        0    1      2
Colorado    3    4      5
```

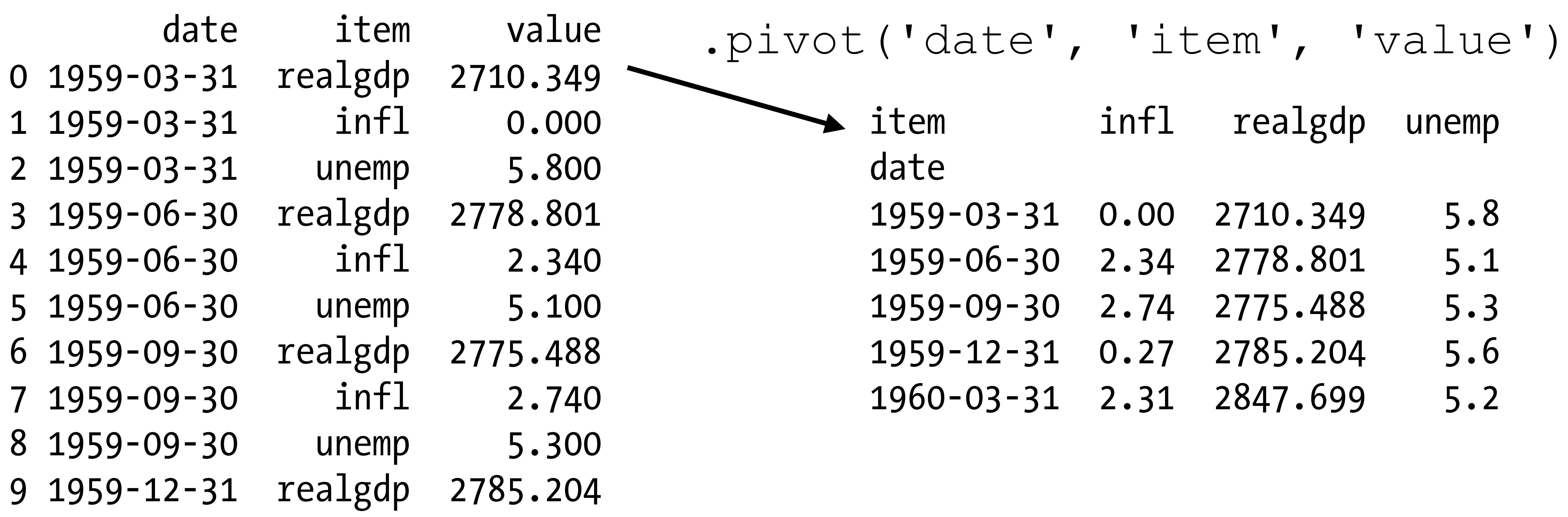
```
state     Ohio  Colorado
number
one          0         3
two          1         4
three        2         5
```

```
state     number
Ohio      one       0
          two       1
          three     2
Colorado  one       3
          two       4
          three     5
```

[W. McKinney, Python for Data Analysis]

Pivot

- Sometimes, we have data that is given in "long" format and we would like "wide" format
- Long format: column names are data values…
- Wide format: more like spreadsheet format
- Example:



```
         date     item     value
0 1959-03-31  realgdp  2710.349
1 1959-03-31     infl     0.000
2 1959-03-31    unemp     5.800
3 1959-06-30  realgdp  2778.801
4 1959-06-30     infl     2.340
5 1959-06-30    unemp     5.100
6 1959-09-30  realgdp  2775.488
7 1959-09-30     infl     2.740
8 1959-09-30    unemp     5.300
9 1959-12-31  realgdp  2785.204
```

```
.pivot('date', 'item', 'value')
```

```
item        infl   realgdp  unemp
date
1959-03-31  0.00  2710.349    5.8
1959-06-30  2.34  2778.801    5.1
1959-09-30  2.74  2775.488    5.3
1959-12-31  0.27  2785.204    5.6
1960-03-31  2.31  2847.699    5.2
```

[W. McKinney, Python for Data Analysis]

Melt

- Turn columns into rows
- One or more columns become rows under a new column (`column`)
- Values become a new column (`value`)
- After melt, data is **molten**
- **Inverse** of pivot

row	a	b	c
A	1	4	7
B	2	5	8
C	3	6	9

(a) Raw data

row	column	value
A	a	1
B	a	2
C	a	3
A	b	4
B	b	5
C	b	6
A	c	7
B	c	8
C	c	9

(b) Molten data

[H. Wickham, 2014]

Problem: Variables stored in both rows & columns

Mexico Weather, Global Historical Climatology Network

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	—	—	—	—	—	—	—	—

[H. Wickham, 2014]

Problem: Variables stored in both rows & columns

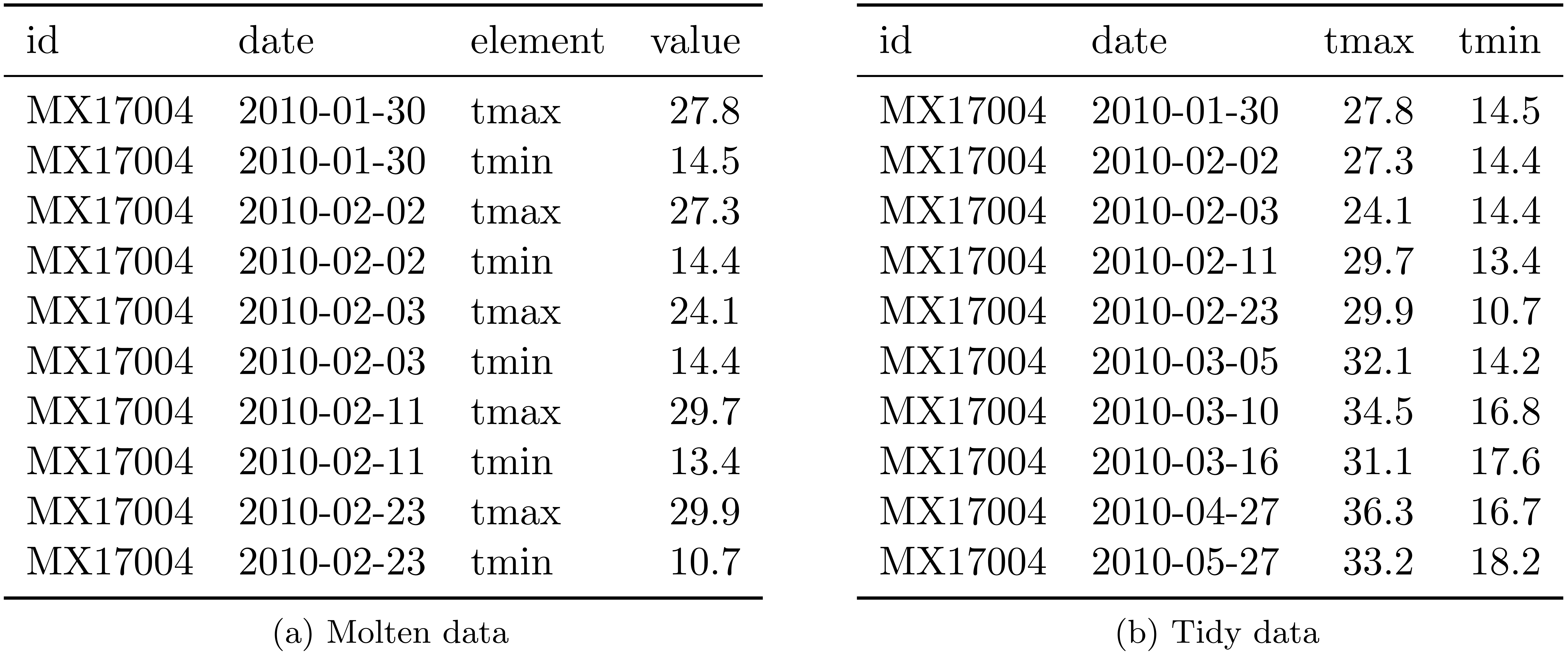
Mexico Weather, Global Historical Climatology Network

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	—	—	—	—	—	—	—	—

Variable in columns: day; Variable in rows: tmax/tmin

[H. Wickham, 2014]

Solution: Melting + Pivot



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

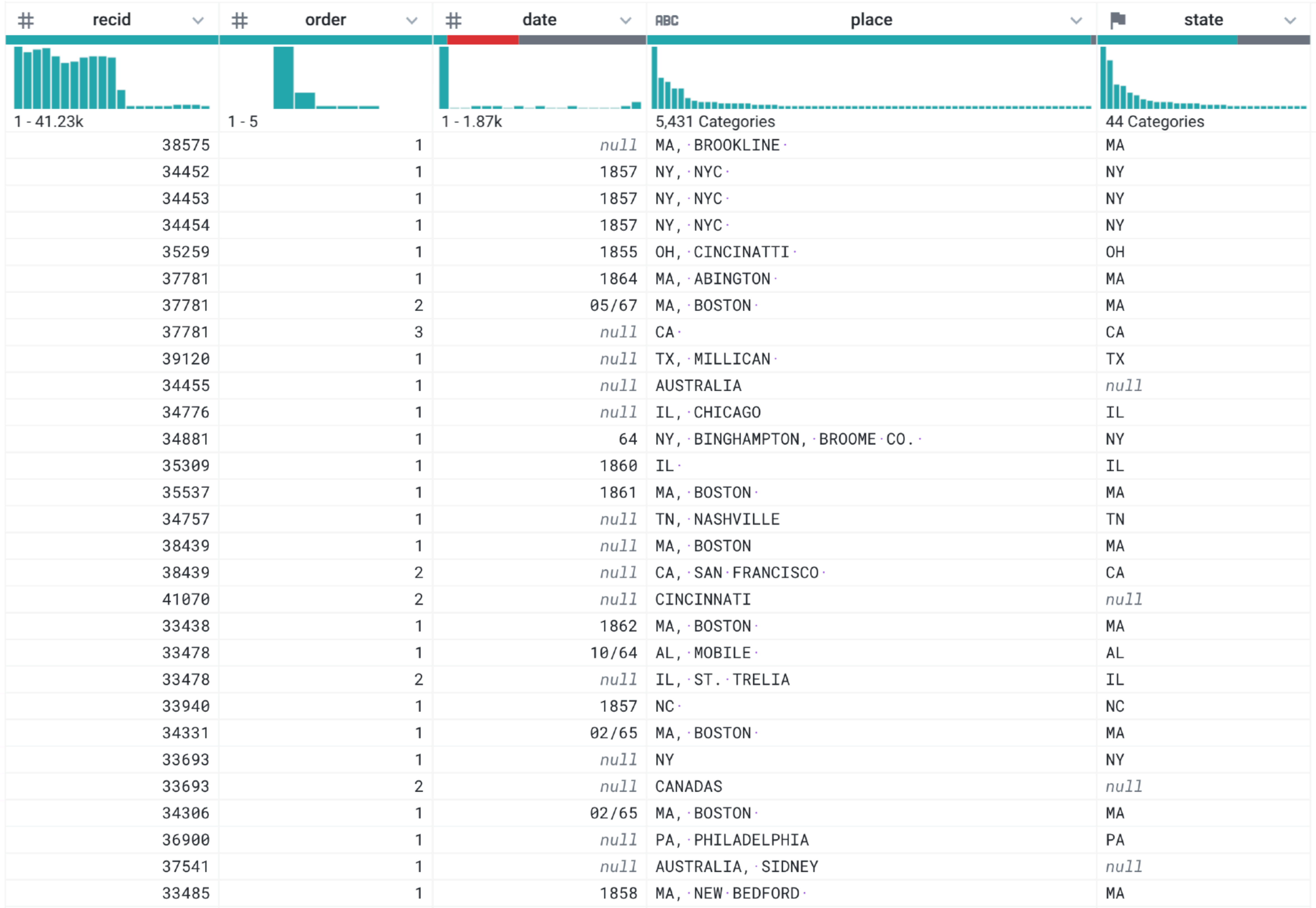
id	date	tmax	tmin
MX17004	2010-01-30	27.8	14.5
MX17004	2010-02-02	27.3	14.4
MX17004	2010-02-03	24.1	14.4
MX17004	2010-02-11	29.7	13.4
MX17004	2010-02-23	29.9	10.7
MX17004	2010-03-05	32.1	14.2
MX17004	2010-03-10	34.5	16.8
MX17004	2010-03-16	31.1	17.6
MX17004	2010-04-27	36.3	16.7
MX17004	2010-05-27	33.2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

(b) Tidy data

[H. Wickham, 2014]

Assignment 3

- Same Info Wanted data
- Data wrangling with
  - Trifacta Wrangler
  - pandas
- For place, date extraction: 2 regexs, don't try to standardize anything, CS680 need to extract place details, date is EC
- Start now!
- Due Wednesday, March 3



# recid	# order	# date	ABC place	state
1 - 41.23k	1 - 5	1 - 1.87k	5,431 Categories	44 Categories
38575	1	*null*	MA, BROOKLINE	MA
34452	1	1857	NY, NYC	NY
34453	1	1857	NY, NYC	NY
34454	1	1857	NY, NYC	NY
35259	1	1855	OH, CINCINATTI	OH
37781	1	1864	MA, ABINGTON	MA
37781	2	05/67	MA, BOSTON	MA
37781	3	*null*	CA	CA
39120	1	*null*	TX, MILLICAN	TX
34455	1	*null*	AUSTRALIA	*null*
34776	1	*null*	IL, CHICAGO	IL
34881	1	64	NY, BINGHAMPTON, BROOME CO.	NY
35309	1	1860	IL	IL
35537	1	1861	MA, BOSTON	MA
34757	1	*null*	TN, NASHVILLE	TN
38439	1	*null*	MA, BOSTON	MA
38439	2	*null*	CA, SAN FRANCISCO	CA
41070	2	*null*	CINCINNATI	*null*
33438	1	1862	MA, BOSTON	MA
33478	1	10/64	AL, MOBILE	AL
33478	2	*null*	IL, ST. TRELIA	IL
33940	1	1857	NC	NC
34331	1	02/65	MA, BOSTON	MA
33693	1	*null*	NY	NY
33693	2	*null*	CANADAS	*null*
34306	1	02/65	MA, BOSTON	MA
36900	1	*null*	PA, PHILADELPHIA	PA
37541	1	*null*	AUSTRALIA, SIDNEY	*null*
33485	1	1858	MA, NEW BEDFORD	MA

Outline

- Combining Data
- Data Integration
- Data Matching (Entity Resolution)
- Data Fusion (Wednesday)
    - Integrating Conflicting Data: The Role of Source Dependence, X. L. Dong et al., 2009
    - **Quiz** at the beginning of class

Databases

- Databases:
  - Have been around for years
  - Organize data by tables, allow powerful queries
  - Most support concurrency: allowing multiple users to work with the database at once
  - Provide many features to ensure data integrity, security
- Database Management Systems (DBMS): software that manages databases and facilitates adding, updating, and removing data as well as queries over the data
- Main language used to interact with databases:
  Structured Query Language (SQL)

Relational Databases

- A specific model for databases [Codd, 1969]
- Extremely popular, supported by most major DBMS (IBM DB2, SQLServer, mySQL, etc.)
- Consists of **relations** (tables) made up of **tuples** (rows)
- Relations reference each other!
  - Types of relationships: one-to-one, many-to-one, many-to-many
- Each tuple has a **key**; to reference a tuple in another relation, use a **foreign key** in the current relation

Example: Football Game Data

- Data about football games, teams, & players
  - Game is between two Teams
  - Each Team has Players
- For each game, we could specify every player and all of their information… why is this bad?

Example: Football Game Data

- Data about football games, teams, & players
  - Game is between two Teams
  - Each Team has Players
- For each game, we could specify every player and all of their information… why is this bad?
- Normalization: reduce redundancy, keep information that doesn't change separate
- 3 Relations: Team, Player, Game
- Each relation only encodes the data specific to what it represents

Player

Id	Name	Height	Weight

Team

Id	Name	Wins	Losses

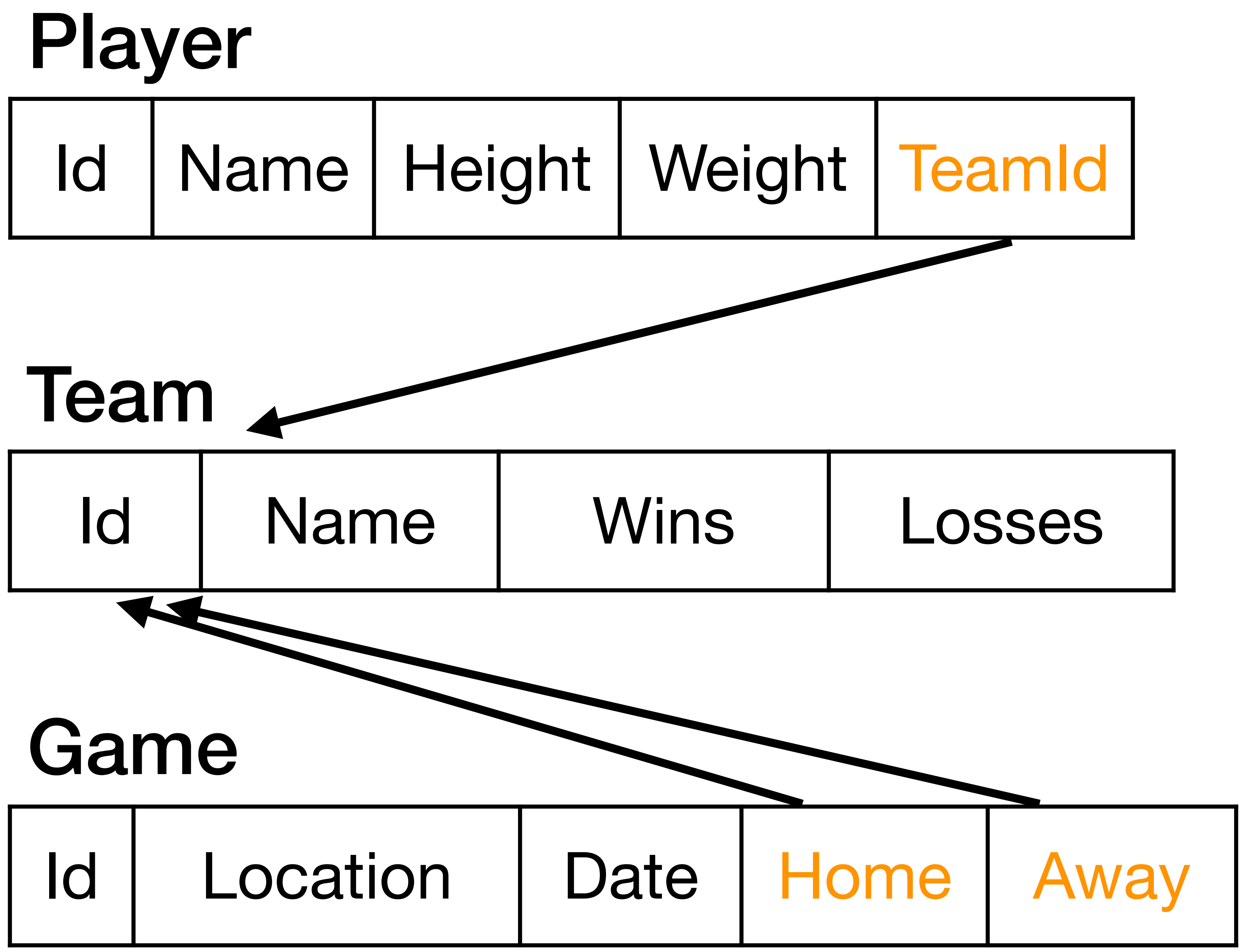
Game

Id	Location	Date

Example: Football Game Data

- Have each game store the id of the home team and the id of the away team (one-to-one)
- Have each player store the id of the team he plays on (many-to-one)

- What happens if a player plays on 2+ teams?



Player

Id	Name	Height	Weight	TeamId

Team

Id	Name	Wins	Losses

Game

Id	Location	Date	Home	Away

How does this relate to pandas?

- DataFrames in pandas are ~relations (tables)
- We may wish to normalize data in a similar manner in pandas
- However, operating on 2+ DataFrames at the same time can be unwieldy, can we merge them together?
- Two potential operations:
  - Have football game data (just the Game table) from 2013, 2014, and 2015 and wish to merge the data into one data frame
  - Have football game data and wish to find the average temperature of the cities where the games were played

Concatenation

- Take two data frames with the same columns and add more rows
- `pd.concat([data-frame-1, data-frame-2, …])`
- Default is to add rows (`axis=0`), but can also add columns (`axis=1`)
- Can also concatenate Series into a data frame.
- `concat` preserves the index so this can be confusing if you have two default indices (0,1,2,3…)—they will appear twice
  - Use `ignore_index=True` to get a 0,1,2…

Merges (aka Joins)

- Need to merge data from one DataFrame with data from another DataFrame
- Example: Football game data merged with temperature data

Game

Id	Location	Date	Home	Away
0	Boston	9/2	1	15
1	Boston	9/9	1	7
2	Cleveland	9/16	12	1
3	San Diego	9/23	21	1

Weather

wId	City	Date	Temp
0	Boston	9/2	72
1	Boston	9/3	68
…	…	…	…
7	Boston	9/9	75
…	…	…	…
21	Boston	9/23	54
…	…	…	…
36	Cleveland	9/16	81

No data for San Diego

Merges (aka Joins)

- Want to join the two tables based on the location and date
- Location and date are the **keys** for the join
- What happens when we have missing data?
- Merges are **ordered**: there is a left and a right side
- Four types of joins:
  - Inner: intersection of keys (match on both sides)
  - Outer: union of keys (if there is no match on other side, still include with NaN to indicate missing data)
  - Left: always have rows from left table (no unmatched right data)
  - Right: like left, but with no unmatched left data

Inner Strategy

Merged

Id	Location	Date	Home	Away	Temp	wId
0	Boston	9/2	1	15	72	0
1	Boston	9/9	1	7	75	7
2	Cleveland	9/16	12	1	81	36

No San Diego entry

Outer Strategy

Merged

Id	Location	Date	Home	Away	Temp	wId
0	Boston	9/2	1	15	72	0
NaN	Boston	9/3	NaN	NaN	68	1
…	…	…	…	…	…	…
1	Boston	9/9	1	7	75	7
NaN	Boston	9/10	NaN	NaN	76	8
…	…	…	…	…	…	…
NaN	Cleveland	9/2	NaN	NaN	61	22
…	…	…	…	…	…	…
2	Cleveland	9/16	12	1	81	36
…	…	…	…	…	…	…
3	San Diego	9/23	21	1	NaN	NaN

Left Strategy

Merged

Id	Location	Date	Home	Away	Temp	wId
0	Boston	9/2	1	15	72	0
1	Boston	9/9	1	7	75	7
2	Cleveland	9/16	12	1	81	36
3	San Diego	9/23	21	1	NaN	NaN

Right Strategy

Merged

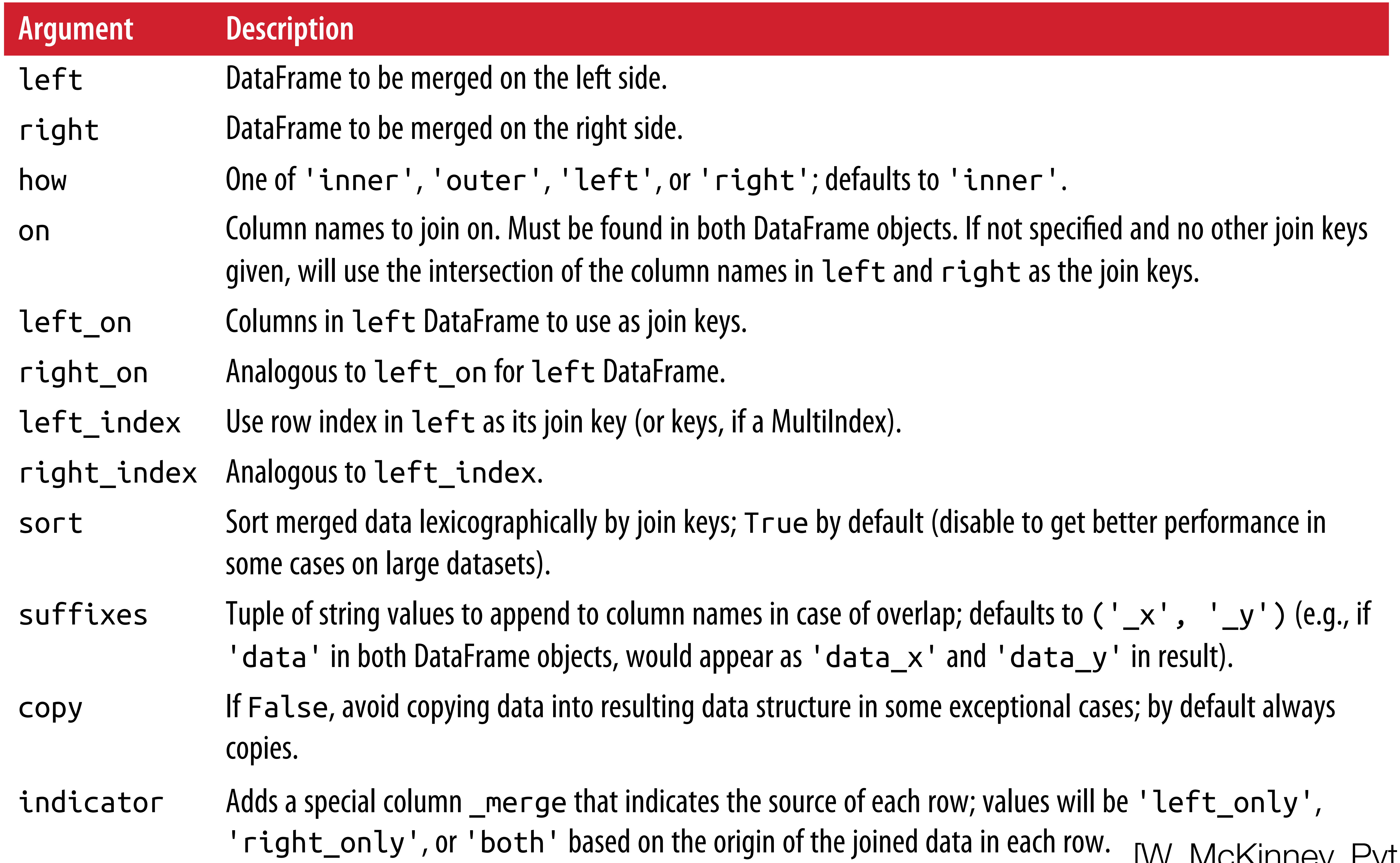
Id	Location	Date	Home	Away	Temp	wId
0	Boston	9/2	1	15	72	0
NaN	Boston	9/3	NaN	NaN	68	1
…	…	…	…	…	…	…
1	Boston	9/9	1	7	75	7
NaN	Boston	9/10	NaN	NaN	76	8
…	…	…	…	…	…	…
NaN	Cleveland	9/2	NaN	NaN	61	22
…	…	…	…	…	…	…
2	Cleveland	9/16	12	1	81	36
…	…	…	…	…	…	…

No San Diego entry

Data Merging in Pandas

- `pd.merge(left, right, …) or left.merge(right, …)`
- Default merge: join on matching column names
- Better: specify the column name(s) to join on via `on` kwarg
  - If column names differ, use `left_on` and `right_on`
  - Multiple keys: use a list
- `how` kwarg specifies type of join `("inner", "outer", "left", "right")`
- Can add suffixes to column names when they appear in both tables, but are not being joined on
- Can also merge using the index by setting `left_index` or `right_index` to `True`

Merge Arguments



Argument	Description
`left`	DataFrame to be merged on the left side.
`right`	DataFrame to be merged on the right side.
`how`	One of `'inner'`, `'outer'`, `'left'`, or `'right'`; defaults to `'inner'`.
`on`	Column names to join on. Must be found in both DataFrame objects. If not specified and no other join keys given, will use the intersection of the column names in `left` and `right` as the join keys.
`left_on`	Columns in `left` DataFrame to use as join keys.
`right_on`	Analogous to `left_on` for `left` DataFrame.
`left_index`	Use row index in `left` as its join key (or keys, if a MultiIndex).
`right_index`	Analogous to `left_index`.
`sort`	Sort merged data lexicographically by join keys; `True` by default (disable to get better performance in some cases on large datasets).
`suffixes`	Tuple of string values to append to column names in case of overlap; defaults to `('_x', '_y')` (e.g., if `'data'` in both DataFrame objects, would appear as `'data_x'` and `'data_y'` in result).
`copy`	If `False`, avoid copying data into resulting data structure in some exceptional cases; by default always copies.
`indicator`	Adds a special column `_merge` that indicates the source of each row; values will be `'left_only'`, `'right_only'`, or `'both'` based on the origin of the joined data in each row.

[W. McKinney, Python for Data Analysis]

Outline

- ~~Combining Data~~
- Data Integration
- Data Matching (Entity Resolution)
- Data Fusion (next Tuesday)
  - Reading Response
  - Integrating Conflicting Data: The Role of Source Dependence, X. L. Dong et al., 2009

Introduction to Data Integration

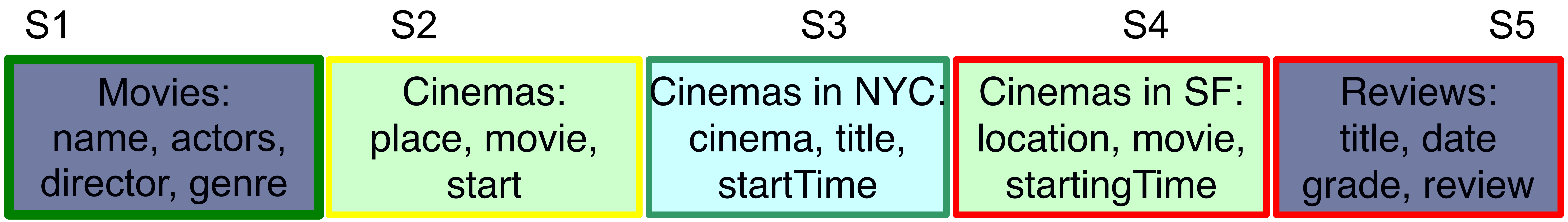
A. Doan, A. Halevy, and Z. Ives

Data Integration

```
select title, startTime
from Movie, Plays
where Movie.title=Plays.movie AND
          location=“New York”  AND
          director=“Woody Allen”
```

Movie: Title, director, year, genre
Actors: title, actor
Plays: movie, location, startTime
Reviews: title, rating, description

Sources S1 and S3 are relevant, sources S4 and S5 are irrelevant, and source S2 is relevant but possibly redundant.



S1

Movies:
name, actors,
director, genre

S2

Cinemas:
place, movie,
start

S3

Cinemas in NYC:
cinema, title,
startTime

S4

Cinemas in SF:
location, movie,
startingTime

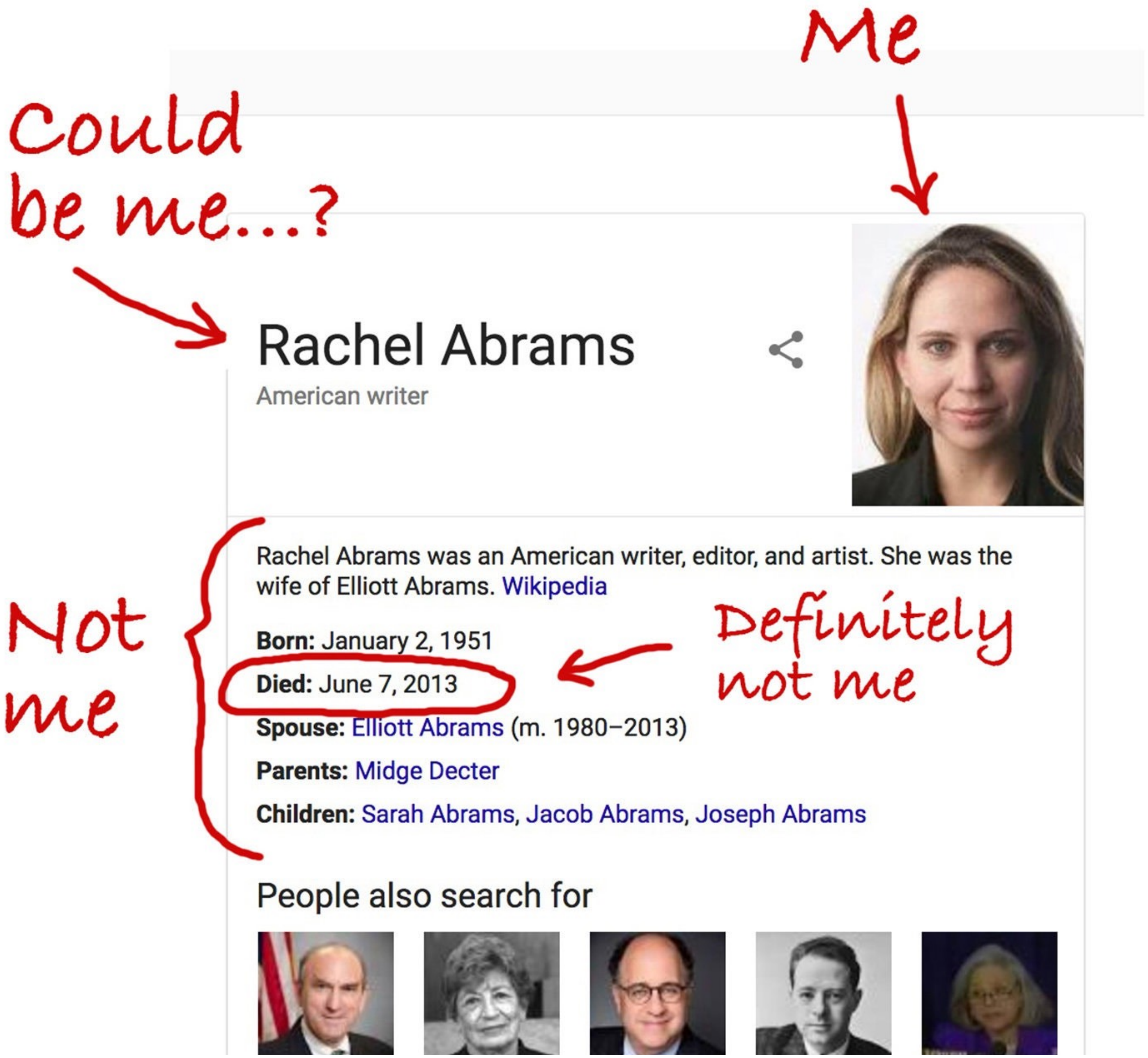
S5

Reviews:
title, date
grade, review

[AH Doan et al., 2012]

Data Matching & Data Fusion

- Google Thinks I'm Dead (I know otherwise.) [R. Abrams, NYTimes, 2017]
- Not only Google, but also Alexa:
  - "Alexa replies that Rachel Abrams is a sprinter from the Northern Mariana Islands (which is true of someone else)."
  - "He asks if Rachel Abrams is deceased, and Alexa responds yes, citing information in the Knowledge Graph panel."



Rachel Abrams

American writer

Rachel Abrams was an American writer, editor, and artist. She was the wife of Elliott Abrams. Wikipedia

Born: January 2, 1951

Died: June 7, 2013

Spouse: Elliott Abrams (m. 1980–2013)

Parents: Midge Decter

Children: Sarah Abrams, Jacob Abrams, Joseph Abrams

People also search for

Data Integration, Data Matching, & Data Fusion

- Data Integration: focus on integrating data from different sources
- Data Matching (aka Entity Resolution aka Record Linkage):
  want to know that two entities (often in different sources) are the same "real" entity
- When sources are orthogonal, no problems
- What happens when two sources provide the same type of information and they **conflict**?
- Data Fusion: create a single object while resolving conflicting values

Record Linkage

P. Christen

Outline

- ~~Combining Data~~
- ~~Data Integration~~
- ~~Data Matching (Entity Resolution)~~
- Data Fusion (Wednesday)
  - Integrating Conflicting Data: The Role of Source Dependence, X. L. Dong et al., 2009
  - **Quiz** at the beginning of class