

Advanced Data Management (CSCI 680/490)

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]

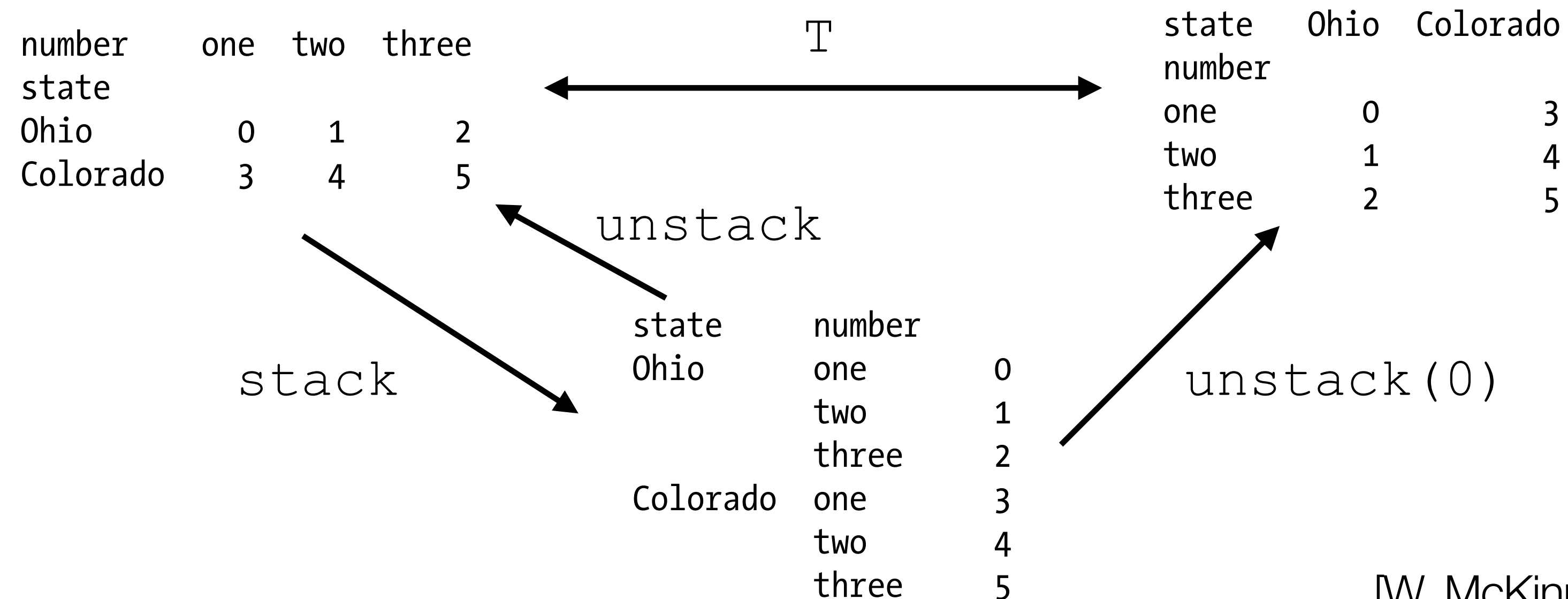
Tidy Data Principles

- **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



[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 dataset as A1 and A2...
- ...but dealing with the full raw data now
- Want to clean and transform using Trifacta and pandas
 - Medium
 - Date cleanup
 - Tags expansion
 - [CSCI 680] Artist Data

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

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

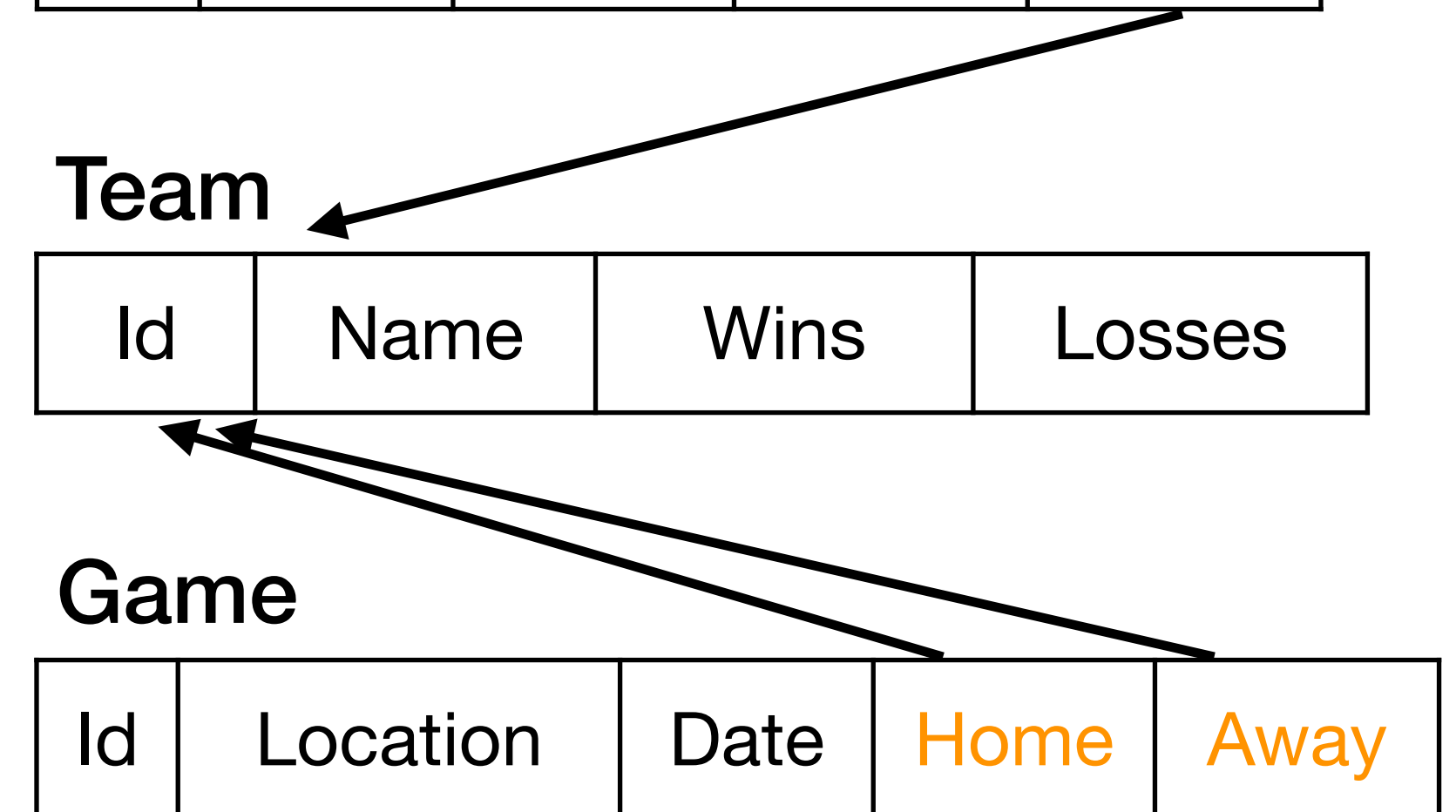
Id	Name	Height	Weight	TeamId
----	------	--------	--------	--------

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

wld	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	wld
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	wld
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	wld
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	wld
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
<code>left</code>	DataFrame to be merged on the left side.
<code>right</code>	DataFrame to be merged on the right side.
<code>how</code>	One of 'inner', 'outer', 'left', or 'right'; defaults to 'inner'.
<code>on</code>	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 <code>left</code> and <code>right</code> as the join keys.
<code>left_on</code>	Columns in <code>left</code> DataFrame to use as join keys.
<code>right_on</code>	Analogous to <code>left_on</code> for <code>left</code> DataFrame.
<code>left_index</code>	Use row index in <code>left</code> as its join key (or keys, if a MultiIndex).
<code>right_index</code>	Analogous to <code>left_index</code> .
<code>sort</code>	Sort merged data lexicographically by join keys; <code>True</code> by default (disable to get better performance in some cases on large datasets).
<code>suffixes</code>	Tuple of string values to append to column names in case of overlap; defaults to <code>('_x' , '_y')</code> (e.g., if 'data' in both DataFrame objects, would appear as 'data_x' and 'data_y' in result).
<code>copy</code>	If <code>False</code> , avoid copying data into resulting data structure in some exceptional cases; by default always copies.
<code>indicator</code>	Adds a special column <code>_merge</code> 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"
```

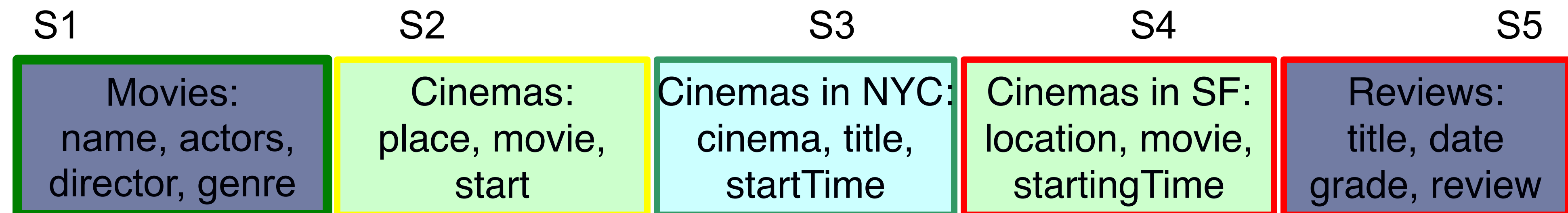
Movie: Title, director, year, genre

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.



[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."

Me ↓

could be me...? →

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

Not me {

Definitely not me ←

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

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

Test 1