Advanced Data Management (CSCI 490/680)

Data Transformation

Dr. David Koop

Foofah's Goal

- Focus on data transformation
- Data transformation tools suffer usability issues:
  - High Skill: familiarity with operations and the effect or their order
  - High Effort: user effort increases as the program becomes longer
- Repetitive and tedious
- Goal: minimize a user's effort and reduce the required background knowledge for data transformation tasks

Getting Lost in Transformations

Bureau of I.A.	
Regional Director	Numbers
Niles C.	Tel: (800)645-8397
	Fax: (907)586-7252
Jean H.	Tel: (918)781-4600
	Fax: (918)781-4604
Frank K.	Tel: (615)564-6500
	Fax: (615)564-6701

Original Table

Split+Delete

Niles C.	Tel	(800)645-8397
	Fax	(907)586-7252
Jean H.	Tel	(918)781-4600
	Fax	(918)781-4604
Frank K.	Tel	(615)564-6500
	Fax	(615)564-6701

Intermediate Table

Unfold

	Tel	Fax
Niles C.	(800)645-8397	
		(615)564-6701
Jean H.	(918)781-4600	
Frank K.	(615)564-6500	

Problem Table

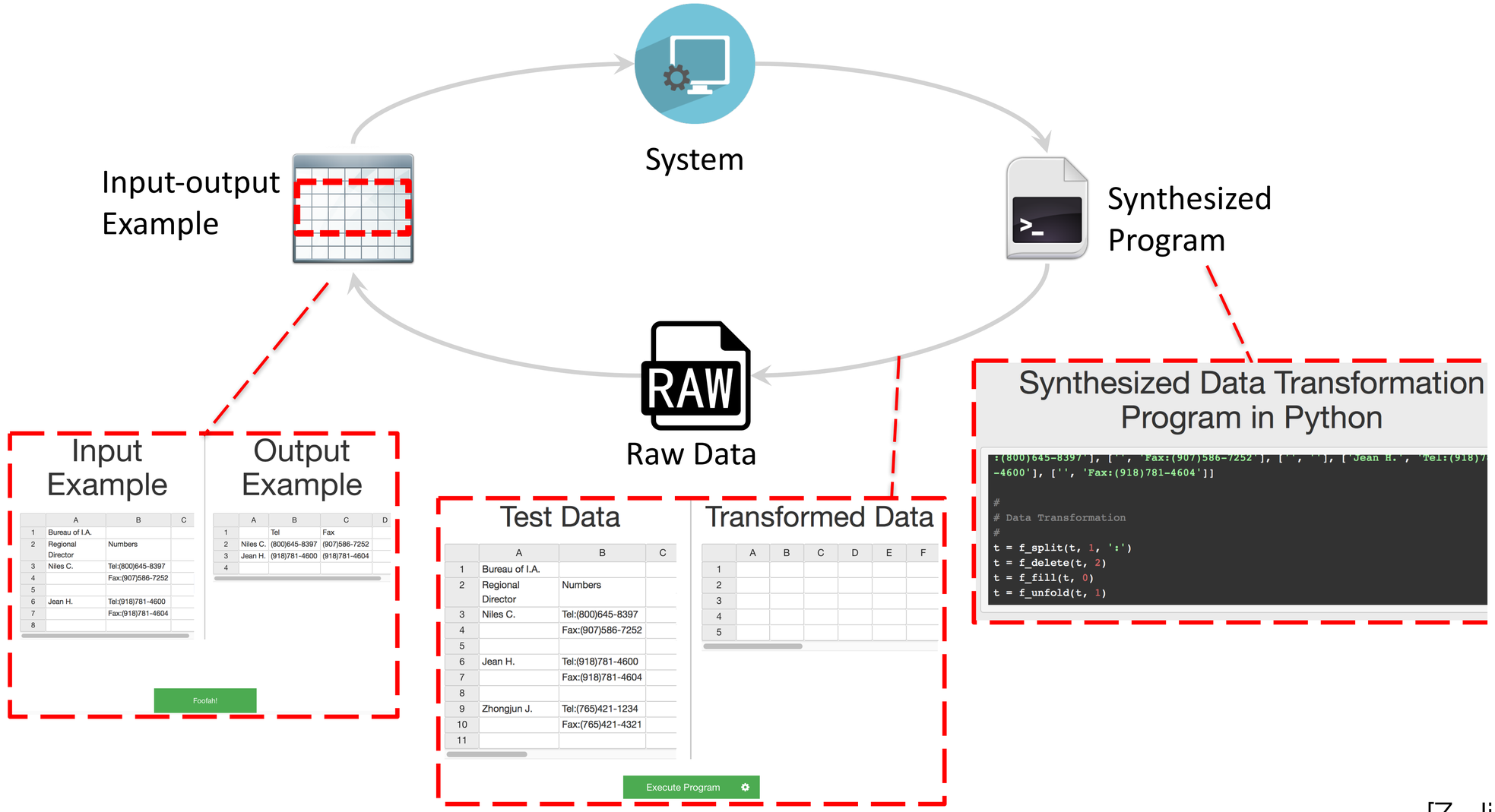
Fill+ Unfold

	Tel	Fax
Niles C.	(800)645-8397	(907)586-7252
Jean H.	(918)781-4600	(918)781-4604
Frank K.	(615)564-6500	(615)564-6701

Desired Solution

[Z. Jin et al., 2017]

Foofah Design: Programming by Example

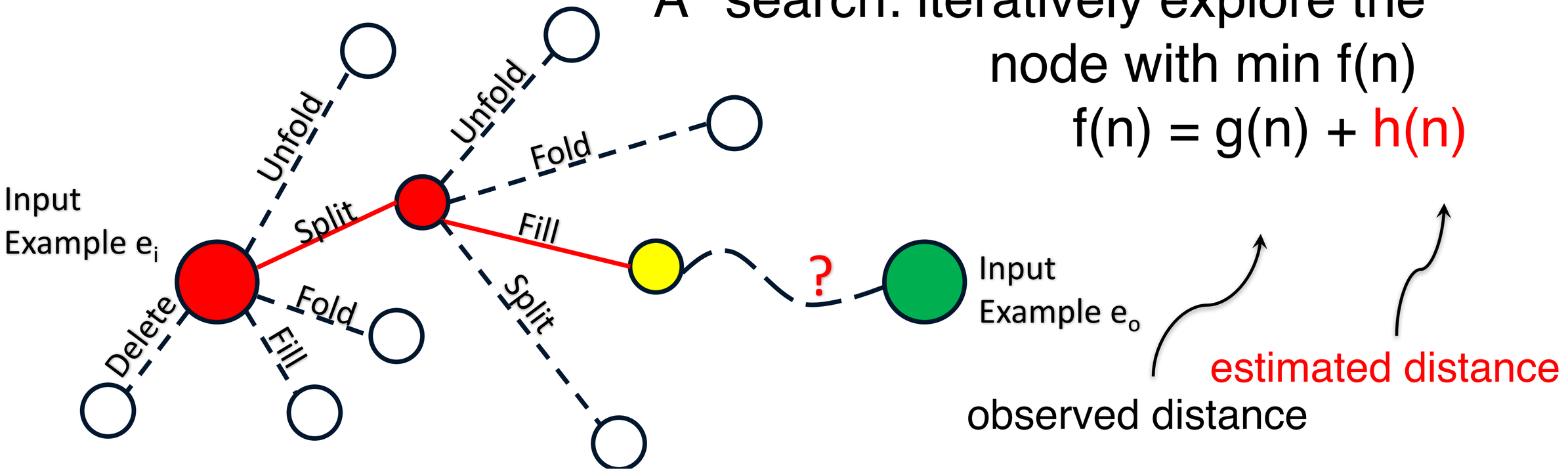


[Z. Jin et al., 2017]

Foofah Solution

A search problem solved by A* algorithm

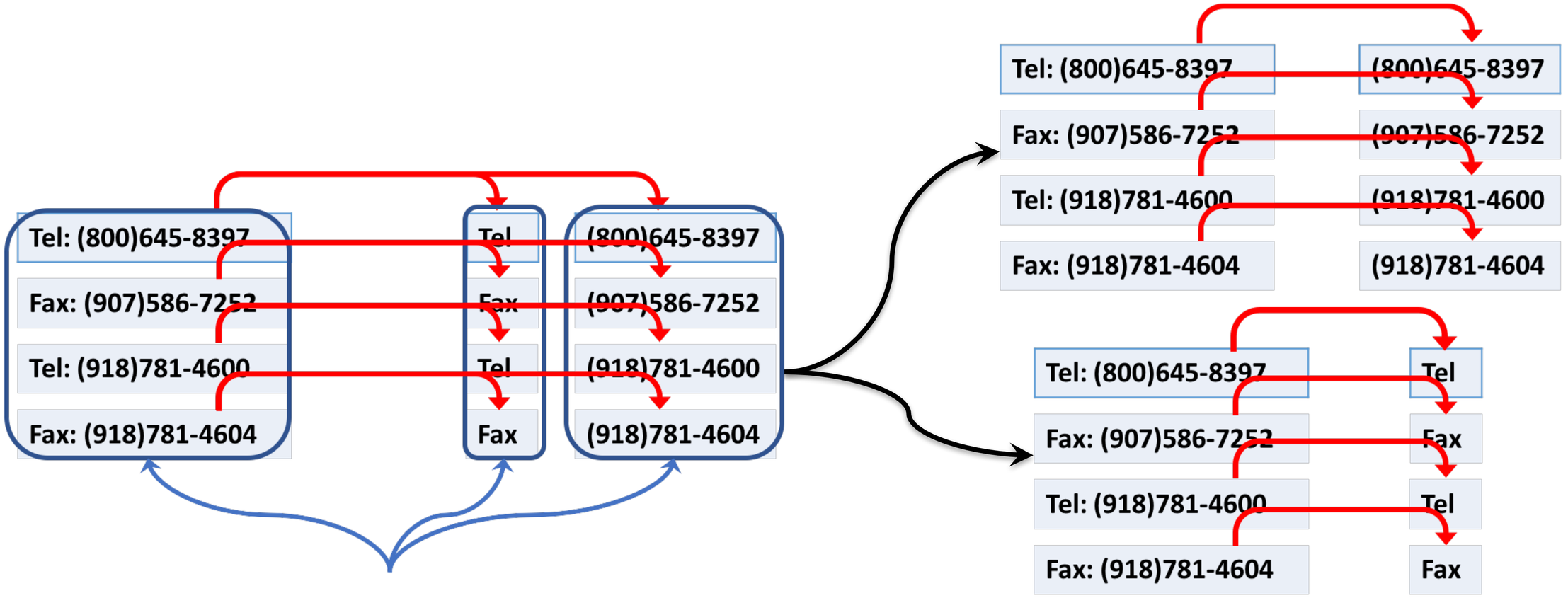
edges: operation
nodes: different views of the data
A* search: iteratively explore the node with min f(n)
f(n) = g(n) + h(n)



[Z. Jin et al., 2017]

Table Edit Distance Batch

Batch the geometrically-adjacent cell-level operations of the same type



8 **`Transform`** operations

2 "batched" **`Transform`** operations

[Z. Jin et al., 2017]

User Study Results

			Wrangler			Foofah		
Test	Complex	≥ 4 Ops	Time	Mouse	Key	Time vs Wrangler	Mouse	Key
PW1	No	No	**104.2**	17.8	11.6	**49.4** ↘**52.6%**	20.8	22.6
PW3 (modified)	No	No	**96.4**	28.8	26.6	**38.6** ↘**60.0%**	14.2	23.6
ProgFromEx13	Yes	No	**263.6**	59.0	16.2	**145.8** ↘**44.7%**	43.6	78.4
PW5	Yes	No	**242.0**	52.0	15.2	**58.8** ↘**75.7%**	31.4	32.4
ProgFromEx17	No	Yes	**72.4**	18.8	11.6	**48.6** ↘**32.9%**	18.2	15.2
PW7	No	Yes	**141.0**	41.8	12.2	**44.4** ↘**68.5%**	19.6	35.8
Proactive1	Yes	Yes	**324.2**	60.0	13.8	**104.2** ↘**67.9%**	41.4	57.0
Wrangler3	Yes	Yes	**590.6**	133.2	29.6	**137.0** ↘**76.8%**	58.6	99.8

[Z. Jin et al., 2017]

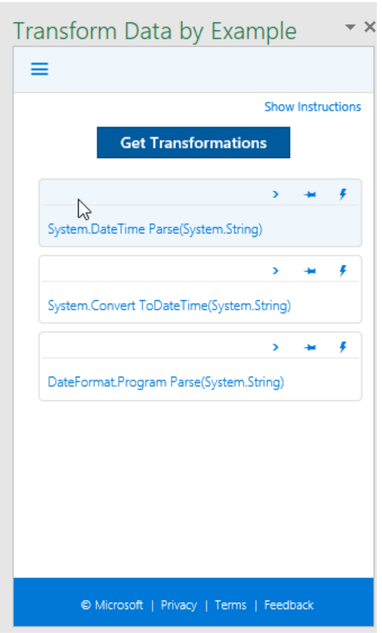
TDE: Transform Data by Example

- Row-to-row translation only
- Search System, GitHub, and StackOverflow for functions
- Given dataset with examples
  - Use L1 from library
  - Compose synthesized programs (L2)
  - Rank best transformations

TDE: Transform Data by Example

C	D
Transaction Date	**output**
Wed, 12 Jan 2011	2011-01-12-Wednesday
Thu, 15 Sep 2011	2011-09-15-Thursday
Mon, 17 Sep 2012	
2010-Nov-30 11:10:41	
2011-Jan-11 02:27:21	
2011-Jan-12	
2010-Dec-24	
9/22/2011	
7/11/2012	
2/12/2012	

C	D
Transaction Date	**output**
Wed, 12 Jan 2011	2011-01-12-Wednesday
Thu, 15 Sep 2011	2011-09-15-Thursday
Mon, 17 Sep 2012	2012-09-17-Monday
2010-Nov-30 11:10:41	2010-11-30-Tuesday
2011-Jan-11 02:27:21	2011-01-11-Tuesday
2011-Jan-12	2011-01-12-Wednesday
2010-Dec-24	2010-12-24-Friday
9/22/2011	2011-09-22-Thursday
7/11/2012	2012-07-11-Wednesday
2/12/2012	2012-02-12-Sunday



[Y. He et al., 2018]

TDE Benchmarks

System	Total cases (239)	FF-GR-Trifacta (46)	Head cases (44)	StackOverflow (49)	BingQL-Unit (50)	BingQL-Other (50)
TDE	**72% (173)**	**91% (42)**	**82% (36)**	**63% (31)**	**96% (48)**	**32% (16)**
TDE-NF	53% (128)	87% (40)	41% (18)	35% (17)	96% (48)	10% (5)
FlashFill	23% (56)	57% (26)	34% (15)	31% (15)	0% (0)	0% (0)
Foofah	3% (7)	9% (4)	2% (1)	4% (2)	0% (0)	0% (0)
DataXFormer-UB	38% (90)	7% (3)	36% (16)	35% (17)	62% (31)	**46% (23)**
System-A	13% (30)	52% (24)	2% (1)	10% (5)	0% (0)	0% (0)
OpenRefine-Menu[8]	4% (9)	13% (6)	2% (1)	4% (2)	0% (0)	0% (0)

- TDE and FlashFill focused on row-to-row transformations
- Foofah considers a wider range of transformations (table reformatting)

[Y. He et al., 2018]

TDE Benchmarks

System	Total cases (239)	FF-GR-Trifacta (46)	Head cases (44)	StackOverflow (49)	BingQL-Unit (50)	BingQL-Other (50)
TDE	**72% (173)**	**91% (42)**	**82% (36)**	**63% (31)**	**96% (48)**	**32% (16)**
TDE-NF	53% (128)	87% (40)	41% (18)	35% (17)	96% (48)	10% (5)
FlashFill	23% (56)	57% (26)	34% (15)	31% (15)	0% (0)	0% (0)
Foofah	3% (7)	9% (4)	2% (1)	4% (2)	0% (0)	0% (0)
DataXFormer-UB	38% (90)	7% (3)	36% (16)	35% (17)	62% (31)	**46% (23)**
System-A	13% (30)	52% (24)	2% (1)	10% (5)	0% (0)	0% (0)
OpenRefine-Menu[8]	4% (9)	13% (6)	2% (1)	4% (2)	0% (0)	0% (0)

- TDE and FlashFill focused on row-to-row transformations
- Foofah considers a wider range of transformations (table reformatting)

[Y. He et al., 2018]

Test 1

- Tuesday, February 18 in class
- Format
  - Multiple Choice
  - Free Response
  - CS680 students will have additional questions
- Coding questions will focus on broad syntax not your memorization of every pandas function
- Concept questions can include discussions of the research papers

Pandas Transformations

- Split: str.split
- Fold/Unfold: stack/unstack
- Merge, join, and concatenate documentation:
  - https://pandas.pydata.org/pandas-docs/stable/merging.html

Tidy Data

- Dataset contain values: quantitative and categorical/qualitative
- Value is either:
  - **variable**: all values that measure the same underlying attribute
  - **observation**: all values measured on the same unit across attributes

[H. Wickham, 2014]

Three Ways to Present the Same Data

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

Initial Data

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

Transpose

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

[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**

[H. Wickham, 2014]

Tidy 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
- Variables also known as dimensions and measures

[H. Wickham, 2014]

Messy Dataset Problems

- Column headers are values, not variable names
- Multiple variables are stored in one column
- Variables are stored in both rows and columns
- Multiple types of observational units are stored in the same table
- A single observational unit is stored in multiple tables

Problem: Column Headers are Values

Income and Religion, Pew Forum

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

[H. Wickham, 2014]

Problem: Column Headers are Values

Income and Religion, Pew Forum

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

Variables: religion, income, frequency

[H. Wickham, 2014]

Solution: Melt Data

- 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]

Solution: Molten Data

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

Original

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

Molten (first 10 rows)

[H. Wickham, 2014]

Melting: Billboard Top Hits

year	artist	track	time	date.entered	wk1	wk2	wk3
2000	2 Pac	Baby Don't Cry	4:22	2000-02-26	87	82	72
2000	2Ge+her	The Hardest Part Of ...	3:15	2000-09-02	91	87	92
2000	3 Doors Down	Kryptonite	3:53	2000-04-08	81	70	68
2000	98^0	Give Me Just One Nig...	3:24	2000-08-19	51	39	34
2000	A*Teens	Dancing Queen	3:44	2000-07-08	97	97	96
2000	Aaliyah	I Don't Wanna	4:15	2000-01-29	84	62	51
2000	Aaliyah	Try Again	4:03	2000-03-18	59	53	38
2000	Adams, Yolanda	Open My Heart	5:30	2000-08-26	76	76	74

Table 7: The first eight Billboard top hits for 2000. Other columns not shown are `wk4`, `wk5`, ..., `wk75`.

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

[Wickham, 2014]

Melting

- Pandas also has a melt function:

```
In [41]: cheese = pd.DataFrame({'first' : ['John', 'Mary'],
   ....:                        'last' : ['Doe', 'Bo'],
   ....:                        'height' : [5.5, 6.0],
   ....:                        'weight' : [130, 150]})
   ....:

In [42]: cheese
Out[42]:
  first  height last  weight
0  John     5.5  Doe     130
1  Mary     6.0   Bo     150

In [43]: cheese.melt(id_vars=['first', 'last'])
Out[43]:
  first last variable  value
0  John  Doe   height    5.5
1  Mary   Bo   height    6.0
2  John  Doe   weight  130.0
3  Mary   Bo   weight  150.0

In [44]: cheese.melt(id_vars=['first', 'last'], var_name='quantity')
Out[44]:
  first last quantity  value
0  John  Doe   height    5.5
1  Mary   Bo   height    6.0
2  John  Doe   weight  130.0
3  Mary   Bo   weight  150.0
```

Problem: Multiple variables stored in one column

Tuberculosis Data, World Health Organization

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	—	—	—

[H. Wickham, 2014]

Problem: Multiple variables stored in one column

Tuberculosis Data, World Health Organization

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	—	—	—

Two variables in columns: age and sex

[H. Wickham, 2014]

Solution: Melting + Splitting

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

[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	—	—	—	—	—	—	—	—

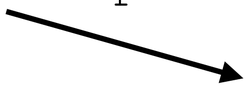
Variable in columns: day; Variable in rows: tmax/tmin

[H. Wickham, 2014]

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]

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]

Problem: Multiple types in one table

Billboard Top Hits

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

[H. Wickham, 2014]

Problem: Multiple types in one table

Billboard Top Hits

Repeated Information

year	artist	time	track	date	week	rank
2000	2 Pac	4:22	Baby Don't Cry	2000-02-26	1	87
2000	2 Pac	4:22	Baby Don't Cry	2000-03-04	2	82
2000	2 Pac	4:22	Baby Don't Cry	2000-03-11	3	72
2000	2 Pac	4:22	Baby Don't Cry	2000-03-18	4	77
2000	2 Pac	4:22	Baby Don't Cry	2000-03-25	5	87
2000	2 Pac	4:22	Baby Don't Cry	2000-04-01	6	94
2000	2 Pac	4:22	Baby Don't Cry	2000-04-08	7	99
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-02	1	91
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-09	2	87
2000	2Ge+her	3:15	The Hardest Part Of ...	2000-09-16	3	92
2000	3 Doors Down	3:53	Kryptonite	2000-04-08	1	81
2000	3 Doors Down	3:53	Kryptonite	2000-04-15	2	70
2000	3 Doors Down	3:53	Kryptonite	2000-04-22	3	68
2000	3 Doors Down	3:53	Kryptonite	2000-04-29	4	67
2000	3 Doors Down	3:53	Kryptonite	2000-05-06	5	66

[H. Wickham, 2014]

Solution: Normalization

id	artist	track	time
1	2 Pac	Baby Don't Cry	4:22
2	2Ge+her	The Hardest Part Of ...	3:15
3	3 Doors Down	Kryptonite	3:53
4	3 Doors Down	Loser	4:24
5	504 Boyz	Wobble Wobble	3:35
6	98ˆ0	Give Me Just One Nig...	3:24
7	A*Teens	Dancing Queen	3:44
8	Aaliyah	I Don't Wanna	4:15
9	Aaliyah	Try Again	4:03
10	Adams, Yolanda	Open My Heart	5:30
11	Adkins, Trace	More	3:05
12	Aguilera, Christina	Come On Over Baby	3:38
13	Aguilera, Christina	I Turn To You	4:00
14	Aguilera, Christina	What A Girl Wants	3:18
15	Alice Deejay	Better Off Alone	6:50

id	date	rank
1	2000-02-26	87
1	2000-03-04	82
1	2000-03-11	72
1	2000-03-18	77
1	2000-03-25	87
1	2000-04-01	94
1	2000-04-08	99
2	2000-09-02	91
2	2000-09-09	87
2	2000-09-16	92
3	2000-04-08	81
3	2000-04-15	70
3	2000-04-22	68
3	2000-04-29	67
3	2000-05-06	66

Important: Analysis may require **merging**!

[H. Wickham, 2014]

Problem: One type in many tables

Baby Names, Social Security Administration

Popularity in 2016

Rank	Male name	Female name
1	Noah	Emma
2	Liam	Olivia
3	William	Ava
4	Mason	Sophia
5	James	Isabella
6	Benjamin	Mia
7	Jacob	Charlotte
8	Michael	Abigail
9	Elijah	Emily
10	Ethan	Harper

Popularity in 2015

Rank	Male name	Female name
1	Noah	Emma
2	Liam	Olivia
3	Mason	Sophia
4	Jacob	Ava
5	William	Isabella
6	Ethan	Mia
7	James	Abigail
8	Alexander	Emily
9	Michael	Charlotte
10	Benjamin	Harper

[Social Security Administration]

Solution: Melt and Concatenation

Rank	Year	Sex	Name
1	2016	Female	Emma
1	2016	Male	Noah
2	2016	Female	Olivia
2	2016	Male	Liam
3	2016	Female	Ava
3	2016	Male	William

Rank	Year	Sex	Name
1	2015	Female	Emma
1	2015	Male	Noah
2	2015	Female	Olivia
2	2015	Male	Liam
3	2015	Female	Sophia
3	2015	Male	Mason

Melt 2015 and 2016

Rank	Year	Sex	Name
1	2015	Female	Emma
1	2015	Male	Noah
1	2016	Female	Emma
1	2016	Male	Noah
2	2015	Female	Olivia
2	2015	Male	Liam
2	2016	Female	Olivia
2	2016	Male	Liam
3	2015	Female	Sophia
3	2015	Male	Mason
3	2016	Female	Ava
3	2016	Male	William

Concatenate

Using Tidy Data

- Sorting (`sort_values`, `sort_index`)
- Transformation (e.g. Fahrenheit to Celsius)
- Filtering (use Boolean indexing)
- Aggregation

Hierarchical Indexing (Multiple Keys)

- We might have multiple keys to identify a single piece of data
- Example: Football records for each team over multiple years
  - Identify a specific row by **both** the team name and the year
  - Can think about this as a tuple (`<team_name>, <year>`)
- pandas supports this via hierarchical indexing (`MultiIndex`)
- display mirrors the hierarchical nature of the data

Example

```
data = [{"W": 11, "L": 5},
        {"W": 6, "L": 10},
        {"W": 12, "L": 4},
         {"W": 8, "L": 8},
        {"W": 2, "L": 14}]
index = [["Boston", "Boston",
          "San Diego", "San Diego",
          "Cleveland"],
         [2007, 2008, 2007,
          2008, 2007]]
df = pd.DataFrame(data,
                  columns=["W", "L"],
                  index=pd.MultiIndex.from_arrays(
                      index, names=('Team', 'Year')))
```

		W	L
Team	**Year**		
Boston	**2007**	11	5
	2008	6	10
San Diego	**2007**	12	4
	2008	8	8
Cleveland	**2007**	2	14

How do we access a row? or slice?

MultiIndex Row Access and Slicing

- `df.loc["Boston", 2007]`
- Remember that `loc` uses the index values, `iloc` uses integers
- **Note**: `df.iloc[0]` gets the first row, **not** `df.iloc[0,0]`
- Can get a subset of the data using partial indices
  - `df.loc["Boston"]` returns both 2007 and 2008 data
- What about slicing?
  - `df.loc["Boston":"Cleveland"]` → ERROR! (Need sorted data)
  - `df = df.sort_index()`
  - `df.loc["Boston":"Cleveland"]` → inclusive!
  - `df.loc[(slice("Boston","Cleveland"),2007),:]`

Reorganizing the MultiIndex

- swaplevel: switch the order of the levels
  - `df = df.swaplevel("Year","Team")`
  - `df.sort_index()`
- Can do summary statistics by level
  - `df.sum(level="Team")`
- Reset the index (back to numbers)
  - `df.reset_index()`
- Promote columns to be the indices
  - `df.set_index(["Team", "Year"])`

		W	L
Team	**Year**		
Boston	**2007**	11	5
	2008	6	10
San Diego	**2007**	12	4
	2008	8	8
Cleveland	**2007**	2	14

Reshaping Data

- Reshape/pivoting are fundamental operations
- Can have a nested index in pandas
- Example: Congressional Districts (Ohio's 1st, 2nd, 3rd, Colorado's 1st, 2nd, 3rd) and associated representative rankings
- Could write this in different ways:

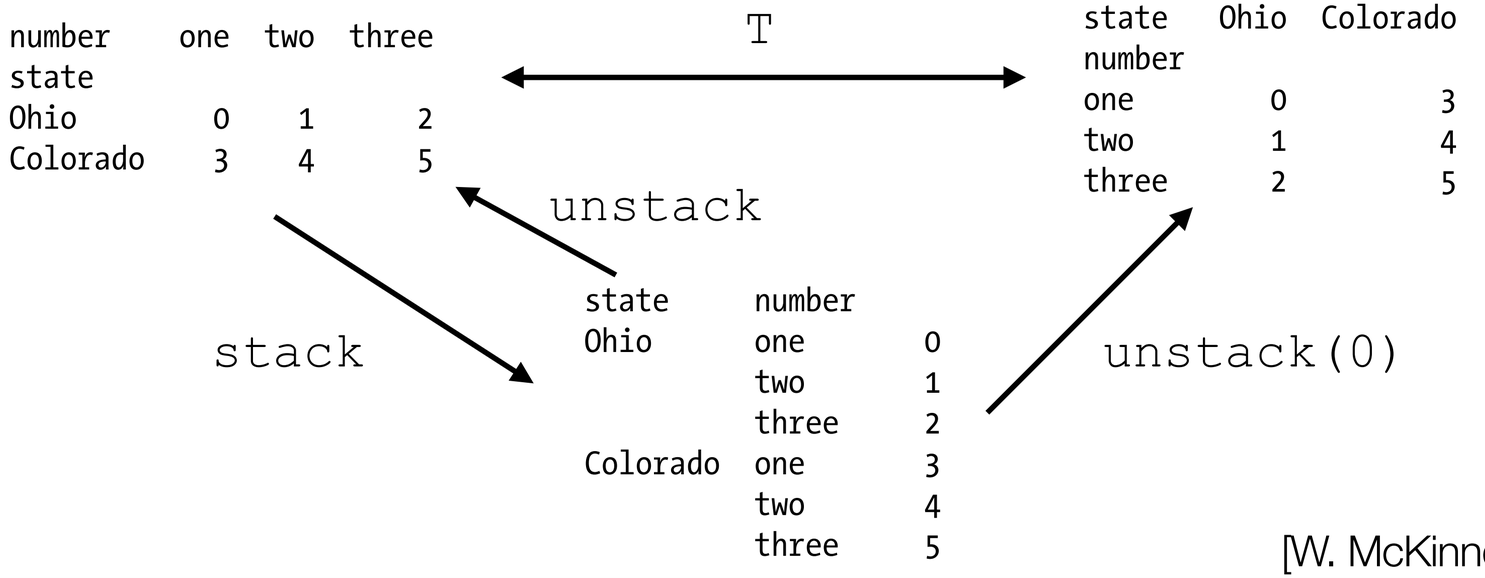
number	one	two	three
state			
Ohio	0	1	2
Colorado	3	4	5

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

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

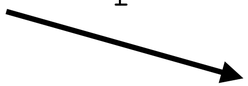


[W. McKinney, Python for Data Analysis]

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]

Baby Names Example