Advanced Data Management (CSCI 640/490)

Data Wrangling & Data Cleaning

Dr. David Koop

Types of Dirty Data Problems

- Separator Issues: e.g. CSV without respecting double quotes
  - `12, 13, "Doe, John", 45`
- Naming Conventions: `NYC` vs. `New York`
- Missing required fields, e.g. key
- Different representations: `2` vs. `two`
- Truncated data: `"Janice Keihanaikukauakahihuliheekahaunaele"` becomes `"Janice Keihanaikukauakahihuliheek"` on Hawaii license
- Redundant records: may be exactly the same or have some overlap
- Formatting issues: `2017-11-07` vs. `07/11/2017` vs. `11/07/2017`

[J. Canny et al.]

Dirty Data: Data Scientist's View

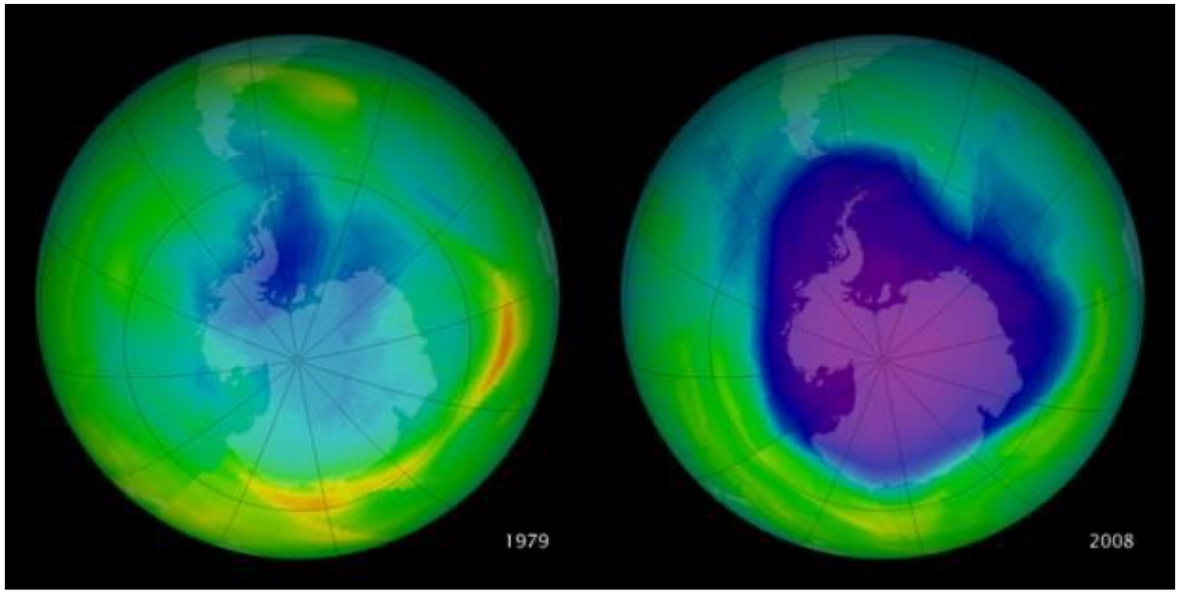
- Combination of:
  - Statistician's View: data has non-ideal samples for model
  - Database Expert's View: missing data, corrupted data
  - Domain Expert's View: data doesn't pass the smell test
- All of the views present problems with the data
- The goal may dictate the solutions:
  - Median value: don't worry too much about crazy outliers
  - Generally, aggregation is less susceptible by numeric errors
  - Be careful, the data may be correct…

[J. Canny et al.]

Be careful how you detect dirty data

- The appearance of a hole in the earth's ozone layer over Antarctica, first detected in 1976, was so unexpected that scientists didn't pay attention to what their instruments were telling them; they thought their instruments were malfunctioning.

  – National Center for Atmospheric Research

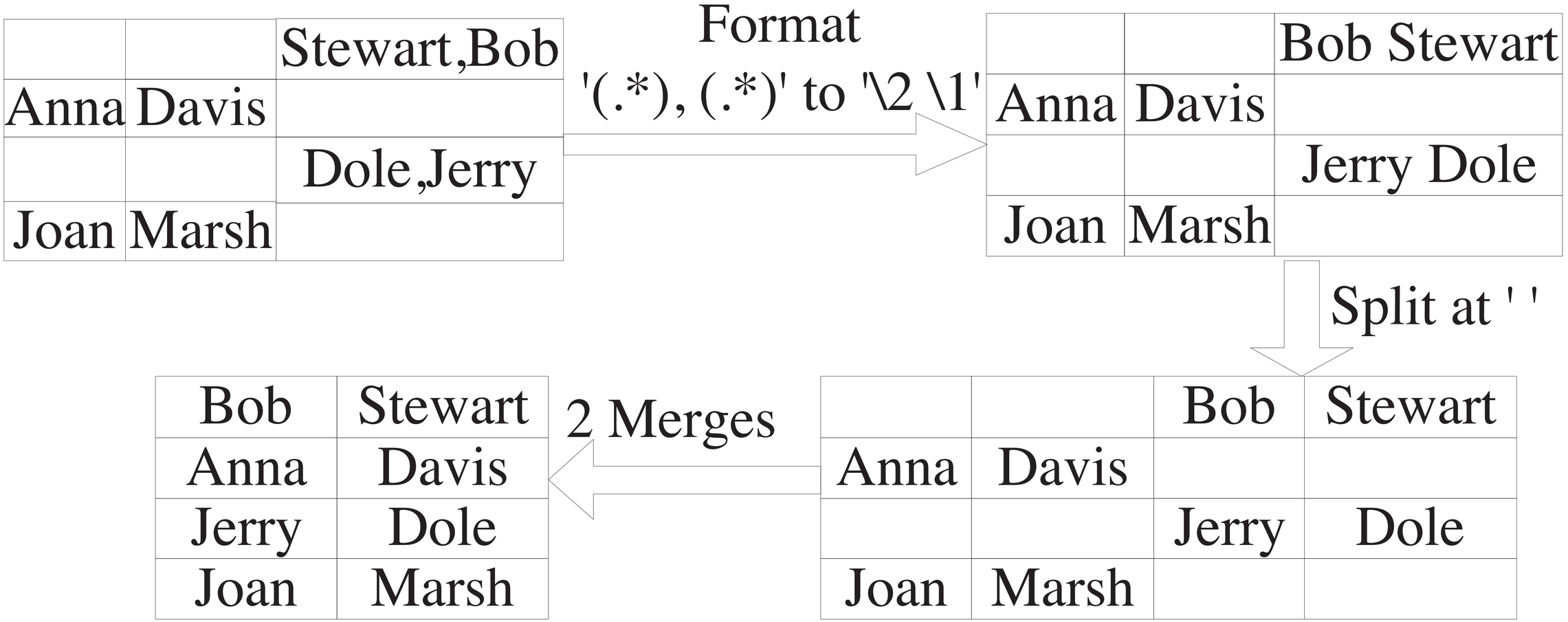


[Wikimedia]

Wrangler

- Data cleaning takes a lot of **time** and **human effort**
- "Tedium is the message"
- Repeating this process on multiple data sets is even worse!
- Solution:
  - interactive interface (mixed-initiative)
  - transformation language with natural language "translations"
  - suggestions + "programming by demonstration"

Potter's Wheel: Example



[V. Raman and J. Hellerstein, 2001]

Potter's Wheel: Transforms

Transform	Definition		
Format	$\phi(R, i, f)$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, f(a_i)) \mid (a_1, \ldots, a_n) \in R\}$
Add	$\alpha(R, x)$	$=$	$\{(a_1, \ldots, a_n, x) \mid (a_1, \ldots, a_n) \in R\}$
Drop	$\pi(R, i)$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n) \mid (a_1, \ldots, a_n) \in R\}$
Copy	$\kappa((a_1, \ldots, a_n), i)$	$=$	$\{(a_1, \ldots, a_n, a_i) \mid (a_1, \ldots, a_n) \in R\}$
Merge	$\mu((a_1, \ldots, a_n), i, j, \text{glue})$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{j-1}, a_{j+1}, \ldots, a_n, a_i \oplus \text{glue} \oplus a_j) \mid (a_1, \ldots, a_n) \in R\}$
Split	$\omega((a_1, \ldots, a_n), i, \text{splitter})$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, \text{left}(a_i, \text{splitter}), \text{right}(a_i, \text{splitter})) \mid (a_1, \ldots, a_n) \in R\}$
Divide	$\delta((a_1, \ldots, a_n), i, \text{pred})$	$=$	$\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, a_i, \text{null}) \mid (a_1, \ldots, a_n) \in R \wedge \text{pred}(a_i)\} \cup$ $\{(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, \text{null}, a_i) \mid (a_1, \ldots, a_n) \in R \wedge \neg\text{pred}(a_i)\}$
Fold	$\lambda(R, i_1, i_2, \ldots i_k)$	$=$	$\{(a_1, \ldots, a_{i_1-1}, a_{i_1+1}, \ldots, a_{i_2-1}, a_{i_2+1}, \ldots, a_{i_k-1}, a_{i_k+1}, \ldots, a_n, a_{i_l}) \mid (a_1, \ldots, a_n) \in R \wedge 1 \leq l \leq k\}$
Select	$\sigma(R, \text{pred})$	$=$	$\{(a_1, \ldots, a_n) \mid (a_1, \ldots, a_n) \in R \wedge \text{pred}((a_1, \ldots, a_n))\}$

Notation: R is a relation with n columns. i, j are column indices and a_i represents the value of a column in a row. x and glue are values. f is a function mapping values to values. $x \oplus y$ concatenates x and y. splitter is a position in a string or a regular expression, $\text{left}(x, \text{splitter})$ is the left part of x after splitting by splitter. pred is a function returning a boolean.

[V. Raman and J. Hellerstein, 2001]

Interface

- Automated Transformation Suggestions
- Editable Natural Language Explanations

- Visual Transformation Previews
- Transformation History

split	split1	split2	split3	split4
	2004	2004	2004	2003
STATE	Participation Rate 2004	Mean SAT I Verbal	Mean SAT I Math	Participation Rate
New York	87	497	510	82
Connecticut	85	515	515	84
Massachusetts	85	518	523	82
New Jersey	83	501	514	85
New Hampshire	80	522	521	75
D.C.	77	489	476	77
Maine	76	505	501	70
Pennsylvania	74	501	502	73
Delaware	73	500	499	73
Georgia	73	494	493	66

split	fold	fold1	value
New York	2004	Participation Rate 2004	87
New York	2004	Mean SAT I Verbal	497
New York	2004	Mean SAT I Math	510
New York	2003	Participation Rate 2003	82
New York	2003	Mean SAT I Verbal	496
New York	2003	Mean SAT I Math	510
Connecticut	2004	Participation Rate 2004	85
Connecticut	2004	Mean SAT I Verbal	515
Connecticut	2004	Mean SAT I Math	515
Connecticut	2003	Participation Rate 2003	84
Connecticut	2003	Mean SAT I Verbal	512

[S. Kandel et al., 2011]

Data Wrangler Demo

- http://vis.stanford.edu/wrangler/app/

Transform Script Import Export

- Split **data repeatedly** on **newline** into **rows**
- Split **split repeatedly** on **','**
- Promote **row 0** to header

Text | Columns | Rows | Table | Clear

Delete **row 7**

Delete **empty rows**

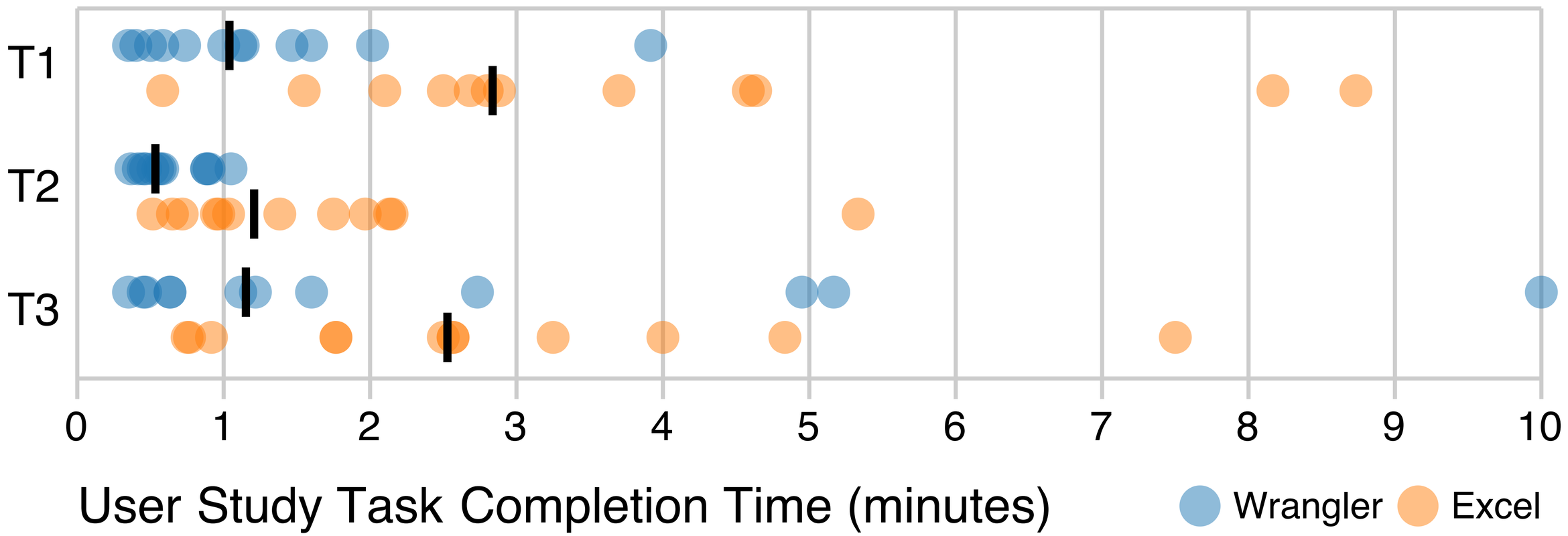
Fill **row 7** by **copying** values from **above**

	Year	Property_crime_rate
0	Reported crime in Alabama	
1		
2	2004	4029.3
3	2005	3900
4	2006	3937
5	2007	3974.9
6	2008	4081.9
7		
8	Reported crime in Alaska	
9		
10	2004	3370.9
11	2005	3615
12	2006	3582

Evaluation

- Compare with Excel
- Tests:
  - Extract text from a single string entry
  - Fill in missing values with estimates
  - Reshape tables
- Allowed users to ask questions about Excel, not Wrangler
- Found significant effect of tool and users found previews and suggestions helpful
- Complaint: No manual fallback, make implications of user choices more obvious for users

Task Completion Times



[S. Kandel et al., 2011]

Courselets

- Make sure to complete these two courselets to learn about **how** to use pandas to do data wrangling and data cleaning
  - using-pandas.ipynb
  - data-cleaning.ipynb

Test 1

- Wednesday, Feb. 28
- In-class, 9:30-10:45am
- Format:
  - Multiple Choice
  - Free Response

Assignment 3

- Same Met Art dataset
- Data Wrangling
  - Using OpenRefine
  - Using pandas

Data Formats

- CSV
  - Text
  - No type information
- JSON
  - Text, Hierarchical
  - Limited type information
- Parquet
  - Binary, Column-oriented
  - Type information
  - Other features: compression

Reading & Writing Data in Pandas

Format	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	Fixed-Width Text File	read_fwf	
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
	MS Excel	read_excel	to_excel
binary	OpenDocument	read_excel	
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	ORC Format	read_orc	
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	SPSS	read_spss	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google BigQuery	read_gbq	to_gbq

[https://pandas.pydata.org/pandas-docs/stable/user_guide/io.html]

Types of arguments for readers

- Indexing: choose a column to index the data, get column names from file or user
- Type inference and data conversion: automatic or user-defined
- Datetime parsing: can combine information from multiple columns
- Iterating: deal with very large files
- Unclean Data: skip rows (e.g. comments) or deal with formatted numbers (e.g. 1,000,345)

Reading/Writing CSV Data with pandas

- Read: `df = pd.read_csv(<path>)`
- Write: `df.to_csv(<path>)`
- Parameters:
  - `sep` (or `delimiter`): the delimiter (`','`, `' '`, `'\t'`, `'\s+'`)
  - `header`: if `None`, no header
  - `names`: list of header names (e.g. if the file has no header)
  - `skiprows`: number of list of lines to skip

Reading/Writing CSV Data with DuckDB

- Importing:
  - `read_csv` method with parameters for delimter, header, etc.
  - `read_csv_auto` automatically **infer** these parameters
  - ```
    CREATE TABLE ontime AS SELECT * FROM
    read_csv_auto('flights.csv');
    ```
- Exporting:
  - Use the `COPY` function
  - `COPY tbl TO 'output.csv' (HEADER, DELIMITER ',');`
    ```

Parquet

- "Open source, column-oriented data file format designed for efficient data storage and retrieval" [parquet.apache.org]
- Available in multiple languages including python
- Binary format
- Column-oriented: can read a column at a time (e.g. from the cloud)
- Self-describing (schema can be embedded)
- Supports compression
- Also supported via Apache Arrow (pyarrow in python) with zero-copy reads

Parquet/CSV Comparison

Dataset	Size on Amazon S3	Query Run time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet format*	**130 GB**	**6.78 seconds**	**2.51 GB**	**$0.01**
Savings / Speedup	**87% less with Parquet**	**34x faster**	**99% less data scanned**	**99.7% savings**

Dataset	Columns	Size on Amazon S3	Data scanned	Cost (1TB = $5)
Data stored as CSV file	4	4TB	4TB	$20
Data stored as GZIP CSV file	4	1TB	1TB	$5
Data stored as Parquet file	4	1TB	0.25TB	$1.25

[T. Spicer]

Parquet Support

- Pandas:
  - Install pyarrow
  - `df = pd.read_parquet('input.parquet')`
  - `df.to_parquet('output.parquet')`
- DuckDB
  - `CREATE TABLE new_tbl AS SELECT * FROM read_parquet('input.parquet');`
  - `COPY tbl TO 'output.parquet' (FORMAT PARQUET);`

Transform Data by Example

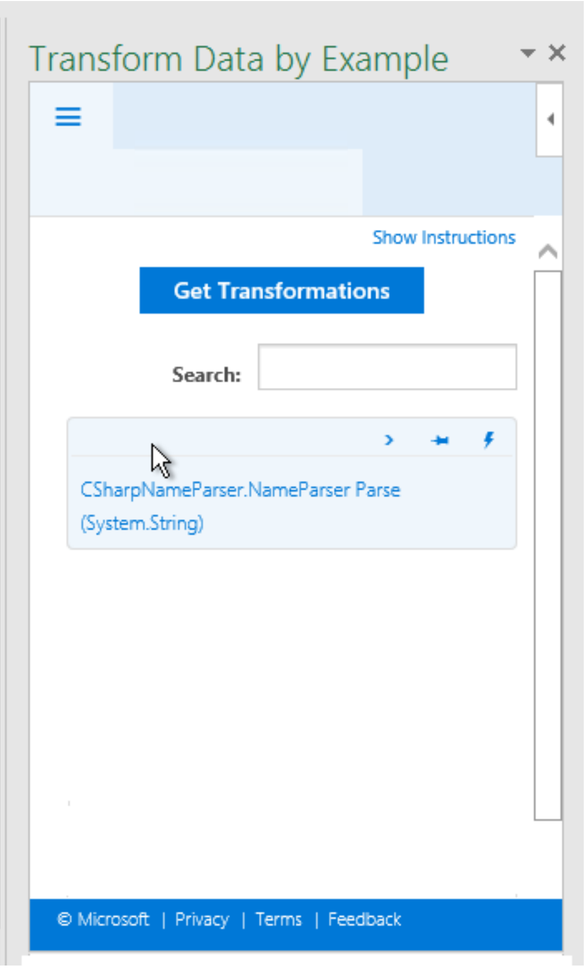
Wrangler

- Have to know what operations to apply
- What about an example-based approach instead?

Microsoft's Transform by Example

TDE: Transform Data by Example

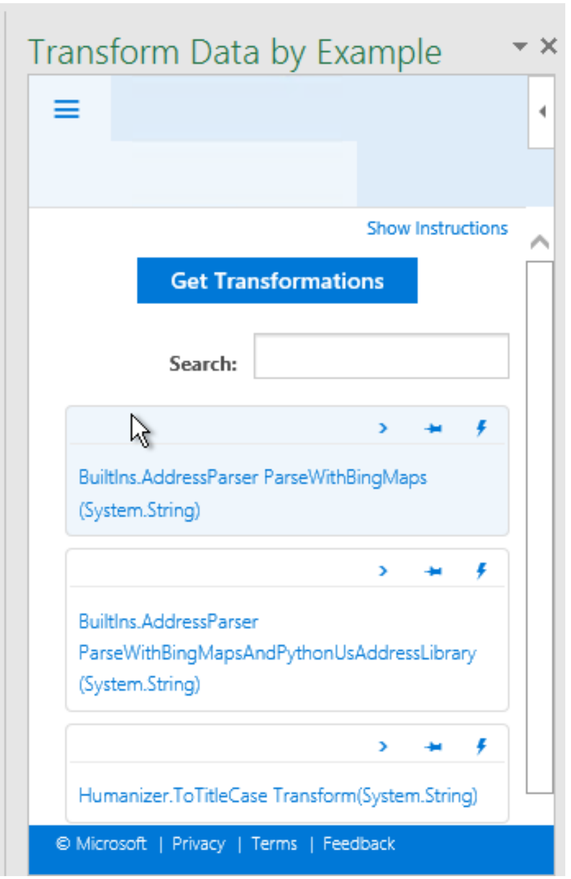
C	D
Customer Name	**Output**
John K. **Doe** Jr.	Doe, John
Mr. **Doe, John**	Doe, John
Jane A. **Smith**	Smith, Jane
MS. **Jane Smith**	Smith, Jane
Smith, Jane	Smith, Jane
Dr **Anthony** R **Von Fange** III	Von Fange, Anthony
Peter Tyson	Tyson, Peter
Dan E. **Williams**	Williams, Dan
James Davis Sr.	Davis, James
James J. **Davis**	Davis, James
Mr. **Donald** Edward **Miller**	Miller, Donald



[Y. He et al., 2018]

TDE: Transform Data by Example

C	D
Address	**Output**
4297 148th Avenue NE L105, Bellevue, WA 98007	Bellevue, WA, 98007
2720 N Mesa St, El Paso, 79902, USA	El Paso, TX, 79902
3524 W Shore Rd APT 1002, Warwick,02886	Warwick, RI, 02886
4740 N 132nd St, Omaha, 68164	Omaha, NE, 68164
10508 Prairie Ln, Oklahoma City	Oklahoma City, OK, 73162
525 1st St, Marysville, WA 95901	Marysville, CA, 95901
211 W Ridge Dr, Waukon,52172	Waukon, IA, 52172
1008 Whitlock Ave NW, Marietta, 30064	Marietta, GA, 30064
602 Highland Ave, Shinnston, 26431	Shinnston, WV, 26431
840 W Star St, Greenville, 27834	Greenville, NC, 27834

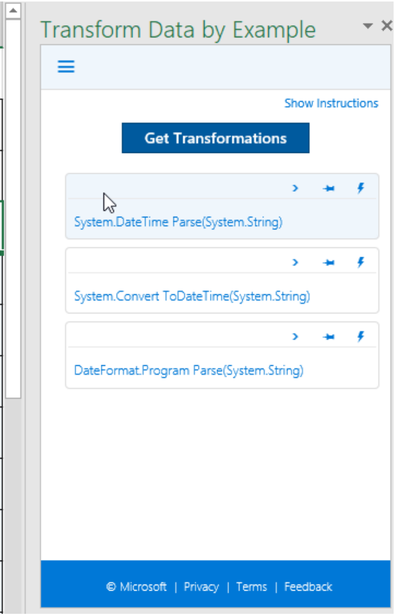


[Y. He et al., 2018]

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: Synthesized Function

Input Examples
Wed, 12 Jan 2011
Thu, 15 Sep 2011
Mon, 17 Sep 2012
2010-Nov-30 21:10:41
2011-Jan-11 02:27:21
2011-Jan-12

f

Return Object Dump					
Year	Month	Day	Day-of-week	Day-of-Year	...
2011	01	12	Wednesday	12	...
2011	09	15	Thursday	258	...
2012	09	17	Monday	261	...
2010	11	30	Tuesday	334	...
2011	01	11	Tuesday	11	...
2011	01	12	Wednesday	12	...

Member method result dump				
ToLongDateString()	ToTimeStr()	ToUTC()	ToBinary()	...
Wednesday, January 12, 2011	12:00:00 AM	...	...	...
Thursday, September 15, 2011	12:00:00 AM	...	...	...
Monday, September 17, 2012	12:00:00 AM	...	...	...
Tuesday, November 30, 2010	09:10:41 PM	...	...	...
Tuesday, January 11, 2011	02:27:21 AM	...	...	...
Wednesday, January 12, 2012	12:00:00 AM	...	...	...

Synthesize

Desired Output
2011-01-12 (Wed)
2011-09-15 (Thu)

[Y. He et al., 2018]

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 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]

Trifacta's Transform by Example

Transform by Pattern (TBP)

- Focus on non-technical users
- More general than Transform by Example
- No need for paired examples
- Use Cases:
  - Auto-Unify: Unify data in different formats
  - Auto-Repair: Fix data quality issues
- Example (Auto-Unify):
  - P_S = `<letter>{3}. <digit>{2}, <digit>{4}`
  - P_T = `<digit>{4}-<digit>{2}-<digit>{2}`

S-timestamp	S-phone	S-coordinates
2019-12-23	(425) 882-8080	(38°57'N, 95°15'W)
2019-12-24	(425) 882-8080	(38°61'N, 95°21'W)
2019-12-23	(206) 876-1800	(39°19'N, 95°18'W)
2019-12-24	(206) 876-1800	(39°26'N, 95°23'W)
2019-12-23	(206) 903-8010	(39°42'N, 96°38'W)

R-timestamp	R-phone	R-coordinates
Nov. 16 2019	650-853-1300	N37°31′ W122°14′
Nov. 17 2019	650-853-1300	N37°18′ W122°19′
Nov. 16 2019	425-421-1225	N37°48′ W122°17′
Nov. 17 2019	425-421-1225	N37°60′ W123°08′
Nov. 16 2019	650-253-0827	N37°01′ W123°72′

[Jin et al.]

TBP Use Cases

- Auto-Unify

S-timestamp	S-phone	S-coordinates
2019-12-23	(425) 882-8080	(38°57'N, 95°15'W)
2019-12-24	(425) 882-8080	(38°61'N, 95°21'W)
2019-12-23	(206) 876-1800	(39°19'N, 95°18'W)
2019-12-24	(206) 876-1800	(39°26'N, 95°23'W)
2019-12-23	(206) 903-8010	(39°42'N, 96°38'W)

R-timestamp	R-phone	R-coordinates
Nov. 16 2019	650-853-1300	N37°31' W122°14'
Nov. 17 2019	650-853-1300	N37°18' W122°19'
Nov. 16 2019	425-421-1225	N37°48' W122°17'
Nov. 17 2019	425-421-1225	N37°60' W123°08'
Nov. 16 2019	650-253-0827	N37°01' W123°72'

- Auto-Repair

Date	Opponents
January 12, 1997	Venezuela
February 12, 1997	Peru
April 2, 1997	Colombia
1997-06-04	United States
1997-06-11	Chile
1997-06-14	Ecuador

(a) EN-Wiki: Dates

Year	Artist	Issue Price (BU)
1989	John Mardon	$16.25
1990	D.J. Craig	$16.75
1991	D.J. Craig	$16.75
1992	Karsten Smith	17.50
1993	Stewart Sherwood	$17.50
1994	Ian D. Sparkes	$17.95

(b) EN-Wiki: Currency values

Women's winner	Time
Anikó Kálovics	2:31:24
Lenah Cheruiyot	2:27:02
Lenah Cheruiyot	2:33.44
Emily Kimuria	2:28.42
Jane Ekimat	2:32.08

(c) EN-wiki:time

#	Original air date[1]
12	March 23, 2008
13	March 30, 2008
14	April 6, 2008
15	13 April 2008
16	20 April 2008

(d) EN-Wiki: Date

[Jin et al.]

TBP Programs and Triples

Table 1: An example repository of TBP programs (P_s, P_t, T), where each line is a TBP program. The first three programs can be used to auto-unify the two tables shown in Figure 2.

TBP-id	Source-pattern (P_s)	Target-pattern (P_t)	(T)
TBP-1	<letter>{3}. <digit>{2}, <digit>{4}	<digit>{4}-<digit>{2}-<digit>{2}	...
TBP-2	(<digit>{3}) <digit>{3}-<digit>{4}	<letter>{3}-<digit>{3}-<digit>{4}	...
TBP-3	(<digit>+°<num>'<letter>{1}, <digit>+°<num>'<letter>{1})	<letter>{1}<digit>+°<num>' <letter>{1}<digit>+°<num>'	...
...	...	...	...
TBP-7	<digit>{4}/<digit>{2}/<digit>{2}	<letter>{3} <digit>{2}	...
TBP-8	<num> kg	<num> lb	...
TBP-9	<num> lb	<num> lb <num> oz	...
...	...	...	...
TBP-15	<num> kg	<num>公斤	...
TBP-16	<letter>+ de <digit>{4}	<digit>{4}	...
...	...	...	...

CCT-id	Input-column (C)	Output-column (C')	Program (T)
CCT-1	(C_1) "Born" = {"02/22/1732", "10/30/1735", ... }	(C'_1) "Date of birth" = {"February 22, 1732", ... }	Listing 1
CCT-2	(C_2) "Date of birth" = {"February 22, 1732", ... }	(C'_2) "Born" = {"02/22/1732", "10/30/1735", ... }	...
CCT-3	(C_3) "Died" = {"02/14/1799", "07/04/1826", ... }	(C'_3) "Date of birth" = {"February 22, 1732", ... }	...
CCT-4	(C_4) "Date" = {"11/01/2019", "12/01/2019", ... }	(C'_4) "Date-2" = {"November 01, 2019", ... }	Listing 1
...	...	...	...
CCT-9	(C_9) "Name" = {"Washington, George", "Adam, John", ... }	(C'_9) "Date of birth" = {"February 22, 1732", ... }	∅
...	...	...	...

[Jin et al.]

Learning TBP Programs

- User Logs
  - Similar to Search Engines
  - (Privacy Issues)
- Tables
  - Find common tables whose rows can be linked
  - Link Wikipedia tables across languages
  - Obtain different data formats and abbreviations that can be used as patterns

[Jin et al.]

TBP Learning from Tables

T_1

Name	#	Born	Died
Washington, George	USA President (1)	02/22/1732	12/14/1799
Adams, John	USA President (2), VP (1)	10/30/1735	07/04/1826
Jefferson, Thomas	USA President (3), VP (2)	04/13/1743	07/04/1826
Madison, James	USA President (4)	03/16/1751	06/28/1836
Monroe, James	USA President (5)	04/28/1758	07/04/1851

T_2

Date of birth	President	Birthplace	State† of birth
February 22, 1732	George Washington	Westmoreland County	Virginia†
October 30, 1735	John Adams	Braintree	Massachusetts†

T_3

30.	George Washington	–	57y, 10d	22.02.1732	14.12.1799
31.	John Quincy Adams	Nat-Rep	57y, 7m, 20d	11.07.1767	23.02.1848
32.	Thomas Jefferson	Dem-Rep	57y, 10m, 18d	13.04.1743	04.07.1826
33.	James Madison	Dem-Rep	57y, 11m, 15d	16.03.1751	28.06.1836
34.	James Monroe	Dem-Rep	58y, 10m, 3d	28.04.1758	04.07.1831

T_4

1.	George Washington	Virginia	Feb. 22, 1732	Dec. 14, 1797
3.	Thomas Jefferson	Virginia	Apr. 13, 1743	July 4, 1826
4.	James Madison	Virginia	Mar. 16, 1751	June 28, 1836
6.	John Quincy Adams	Massachusetts	July 11, 1767	Feb. 23, 1848

T_5

	Name and (party)[1]	Term	State of birth	Born	Died	Religion[2]	Age at inaug.	Age at death
1.	Washington (F)[3]	1789–1797	Va.	2/22/1732	12/14/1799	Episcopalian	57	67
2.	J. Adams (F)	1797–1801	Mass.	10/30/1735	7/4/1826	Unitarian	61	90

T_6

PRESIDENT	BIRTH DATE	BIRTH PLACE	DEATH DATE	LOCATION OF DEATH
George Washington	Feb 22, 1732	Westmoreland Co., Va.	Dec 14, 1799	Mount Vernon, Va.
John Adams	Oct 30, 1735	Quincy, Mass.	July 4, 1826	Quincy, Mass.

[Jin et al.]

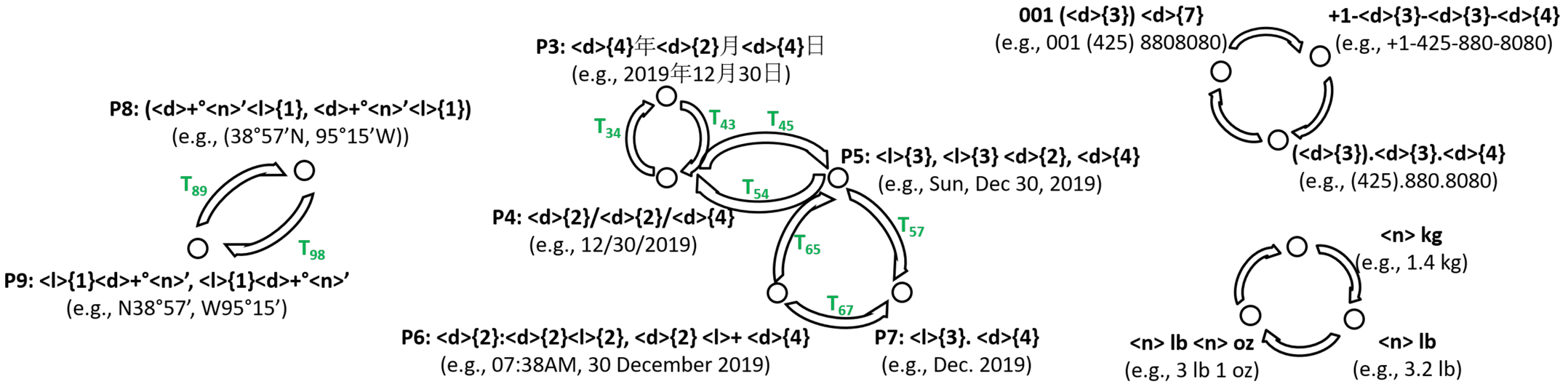
Generating Patterns

- Generate potential regex patterns
- Want more general patterns
- (`<digits>/<digits>/<digits>` vs. `<digits>/<digits>/17<digits>`)
- Can be too general: `<num><symbol><num><symbol><num>`
- Want high "coverage" and high "accuracy"
-

[Jin et al.]

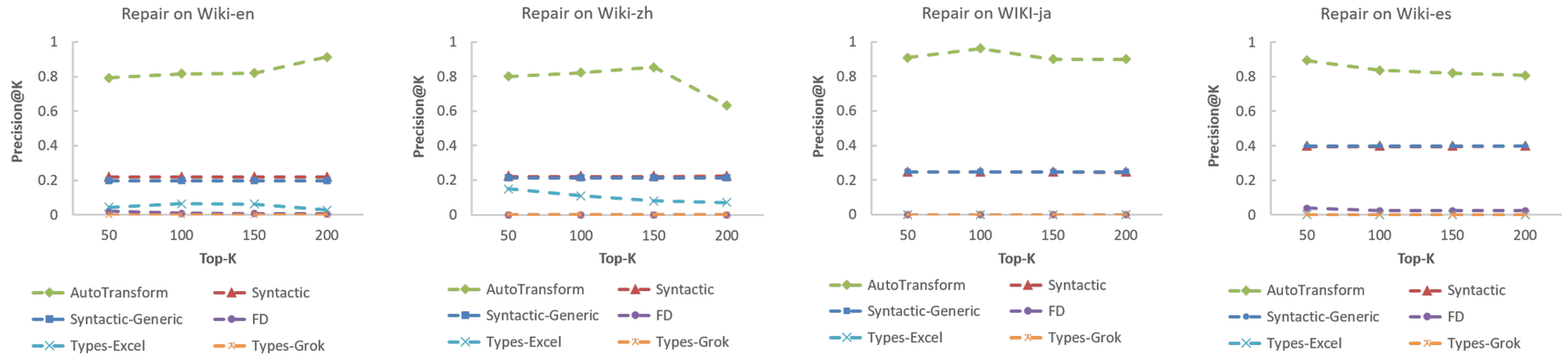
Graph Pattern Relationships

- Lossless inverses: can go back and forth
- Triangular equivalent programs: applying one transformation on a column matches the output of apply two other transformations in sequence



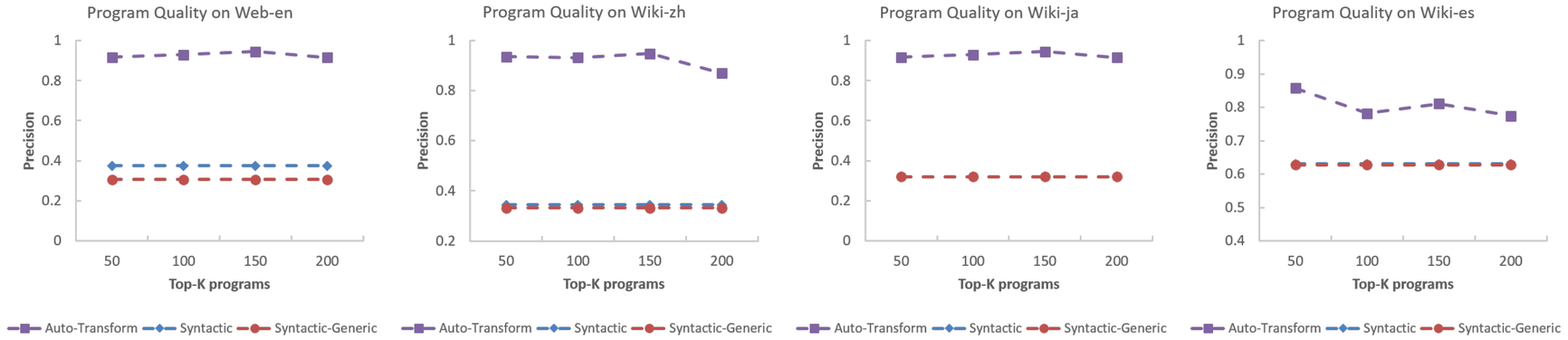
[Jin et al.]

Experiment Results



(a) Wiki-en (b) Wiki-zh (c) Wiki-ja (d) Wiki-es

Figure 10: Quality of repairs on Wiki-en, Wiki-zh, Wiki-ja, Wiki-es, using TBP programs learned from corresponding corpus.



(a) Web-en (b) Wiki-zh (c) Wiki-ja (d) Wiki-es

Figure 11: Quality of TBP programs produced on Web-en, Wiki-zh, Wiki-ja, Wiki-es, respectively.

[Jin et al.]

Data Cleaning

Data Cleaning Types

- How can statistical techniques improve efficiency or reliability of data cleaning? (Data Cleaning **with** Statistics)
  - Example: Trifacta
- How how can we improve the reliability of statistical analytics with data cleaning? (Data Cleaning **for** Statistics)
  - Example: SampleClean

[D. Haas et al., 2016]

Misconceptions about Data Cleaning

- Surveyed Technology Professionals
- The end goal of data cleaning is clean data
  - "We typically clean our data until the desired analytics works without error."
- Data cleaning is a sequential operation
  - "[It's an] iterative process, where I assess biggest problem, devise a fix, re-evaluate. It is dirty work."
- Data cleaning is performed by one person
  - "There are often long back and forths with senior data scientists, devs, and the business units that provided the data on data quality."

[D. Haas et al., 2016]

Misconceptions about Data Cleaning

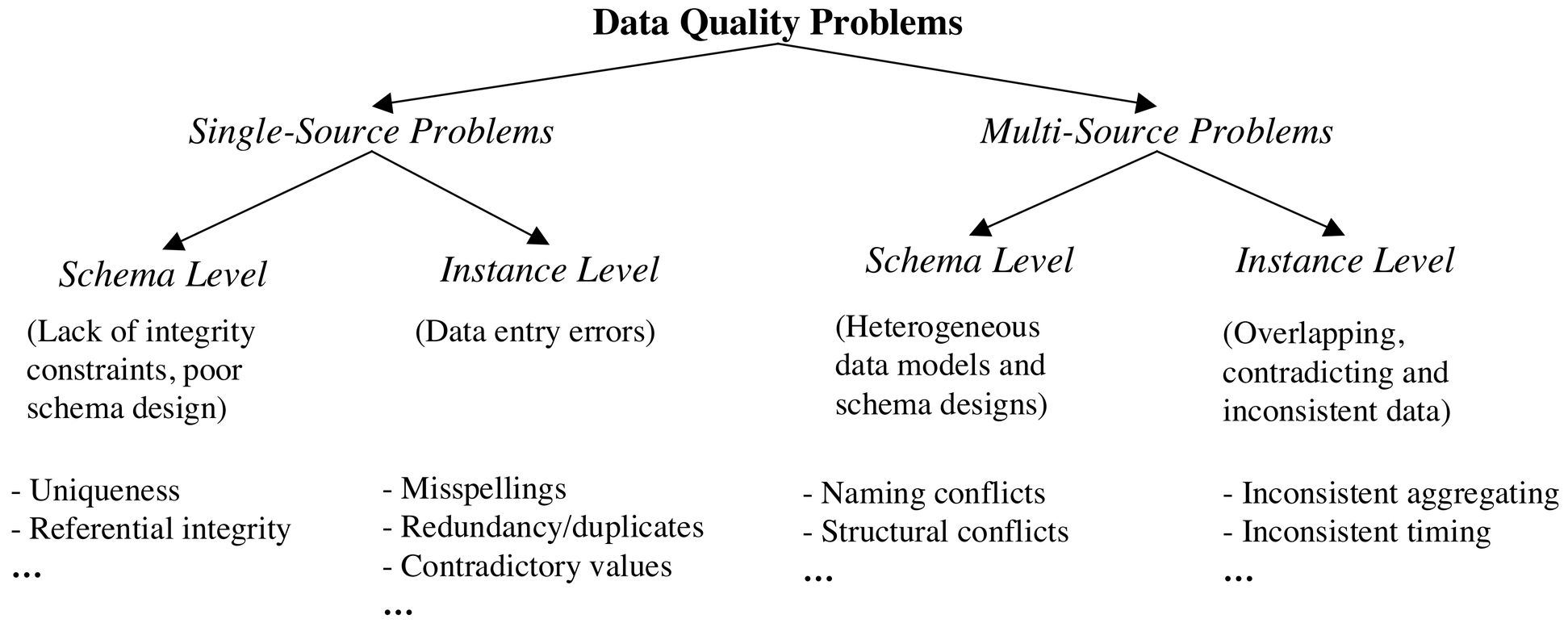
- Data quality is easy to evaluate
  - "I wish there were a more rigorous way to do this but we look at the models and guess if the data are correct"
  - "Other than common sense we do not have a procedure to do this"
  - "Usually [a data error] is only caught weeks later after someone notices."

[D. Haas et al., 2016]

Data Cleaning

- Two key tasks:
  - Error Detection
  - Data Repairing

Classifying Data Quality Problems



[E. Rahm & H. H. Do, 2000]

Single-Source Schema Problems

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Illegal values	bdate=30.13.70	values outside of domain range
Record	Violated attribute dependencies	age=22, bdate=12.02.70	age = (current date – birth date) should hold
Record type	Uniqueness violation	emp_1=(name="John Smith", SSN="123456") emp_2=(name="Peter Miller", SSN="123456")	uniqueness for SSN (social security number) violated
Source	Referential integrity violation	emp=(name="John Smith", deptno=127)	referenced department (127) not defined

[E. Rahm & H. H. Do, 2000]

Single-Source Instance Problems

Scope/Problem		Dirty Data	Reasons/Remarks
Attribute	Missing values	phone=9999-999999	unavailable values during data entry (dummy values or null)
	Misspellings	city=”Liipzig”	usually typos, phonetic errors
	Cryptic values, Abbreviations	experience=”B”; occupation=”DB Prog.”	
	Embedded values	name=”J. Smith 12.02.70 New York”	multiple values entered in one attribute (e.g. in a free-form field)
	Misfielded values	city=”Germany”	
Record	Violated attribute dependencies	city=”Redmond”, zip=77777	city and zip code should correspond
Record type	Word transpositions	$name_1$= “J. Smith”, $name_2$=”Miller P.”	usually in a free-form field
	Duplicated records	emp_1=(name=”John Smith”,...); emp_2=(name=”J. Smith”,...)	same employee represented twice due to some data entry errors
	Contradicting records	emp_1=(name=”John Smith”, bdate=12.02.70); emp_2=(name=”John Smith”, bdate=12.12.70)	the same real world entity is described by different values
Source	Wrong references	emp=(name=”John Smith”, deptno=17)	referenced department (17) is defined but wrong

[E. Rahm & H. H. Do, 2000]

Multi-Source Schema & Instance Problems

Customer (source 1)

CID	*Name*	*Street*	*City*	*Sex*
11	Kristen Smith	2 Hurley Pl	South Fork, MN 48503	0
24	Christian Smith	Hurley St 2	S Fork MN	1

Client (source 2)

Cno	*LastName*	*FirstName*	*Gender*	*Address*	*Phone/Fax*
24	Smith	Christoph	M	23 Harley St, Chicago IL, 60633-2394	333-222-6542 / 333-222-6599
493	Smith	Kris L.	F	2 Hurley Place, South Fork MN, 48503-5998	444-555-6666

Customers (integrated target with cleaned data)

No	*LName*	*FName*	*Gender*	*Street*	*City*	*State*	*ZIP*	*Phone*	*Fax*	*CID*	*Cno*
1	Smith	Kristen L.	F	2 Hurley Place	South Fork	MN	48503-5998	444-555-6666		11	493
2	Smith	Christian	M	2 Hurley Place	South Fork	MN	48503-5998			24	
3	Smith	Christoph	M	23 Harley Street	Chicago	IL	60633-2394	333-222-6542	333-222-6599		24

[E. Rahm & H. H. Do, 2000]

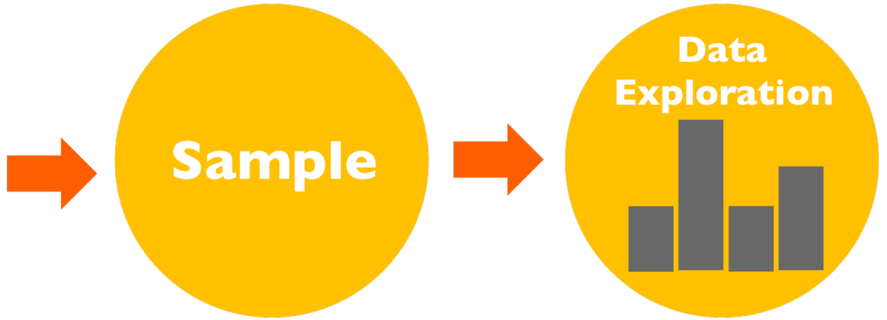
SampleClean (and Variants)

- Dirty Data?
  - Missing Values
  - Duplicate Values
  - Incorrect Values
  - Inconsistent Values
- Estimate query results using a sample of the data
- Two ideas:
  - Direct Estimate
  - Correction

Typical Data Cleaning Steps

Bad Data

Country	UN R/P	UN R/P	World Bank	WB Gini	CIA R/P
Afghanistan	4.1;4.2		27.8	2008	
Albania	7.2	4.8	29	2012	7.2
Algeria	9.6	6.1	35.3		1995,9.6

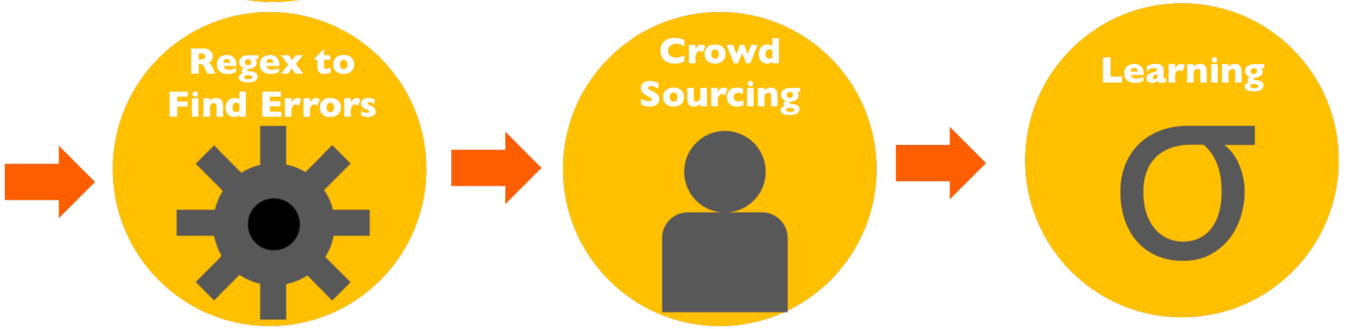


Bad Data

Country	UN R/P	UN R/P	World Bank	WB Gini	CIA R/P
Afghanistan	4.1;4.2		27.8	2008	
Albania	7.2	4.8	29	2012	7.2
Algeria	9.6	6.1	35.3		1995,9.6

Bad Data

Country	UN R/P	UN R/P	World Bank	WB Gini	CIA R/P
Afghanistan	4.1;4.2		27.8	2008	
Albania	7.2	4.8	29	2012	7.2
Algeria	9.6	6.1	35.3		1995,9.6



[sampleclean.org]

Dirty and Cleaned Data

(a) Dirty Data

id	title	pub_year	citation_count
t_1	CrowdDB	11	18
t_2	TinyDB	2005	1569
t_3	YFilter	Feb, 2002	298
t_4	Aqua		106
t_5	DataSpace	2008	107
t_6	CrowdER	2012	1
t_7	Online Aggr.	1997	687
...	...	...	...
t_{10000}	YFilter - ICDE	2002	298

(b) Cleaned Sample

id	title	pub_year	citation_count	#dup
t_1	CrowdDB	**2011**	**144**	**2**
t_2	TinyDB	2005	1569	1
t_3	YFilter	**2002**	298	**2**
t_4	Aqua	**1999**	106	1
t_5	DataSpace	2008	107	1
t_6	CrowdER	2012	**34**	1
t_7	Online Aggr.	1997	687	**3**

[J. Wang et al., 2014]

Two Sources of Error

- Approximate Query Processing (AQP): Don't process the entire dataset, but use samples to get an approximate result
- Now add dirty data
- Two sources of error:

If R is dirty, then there is a true relation R_{clean}.

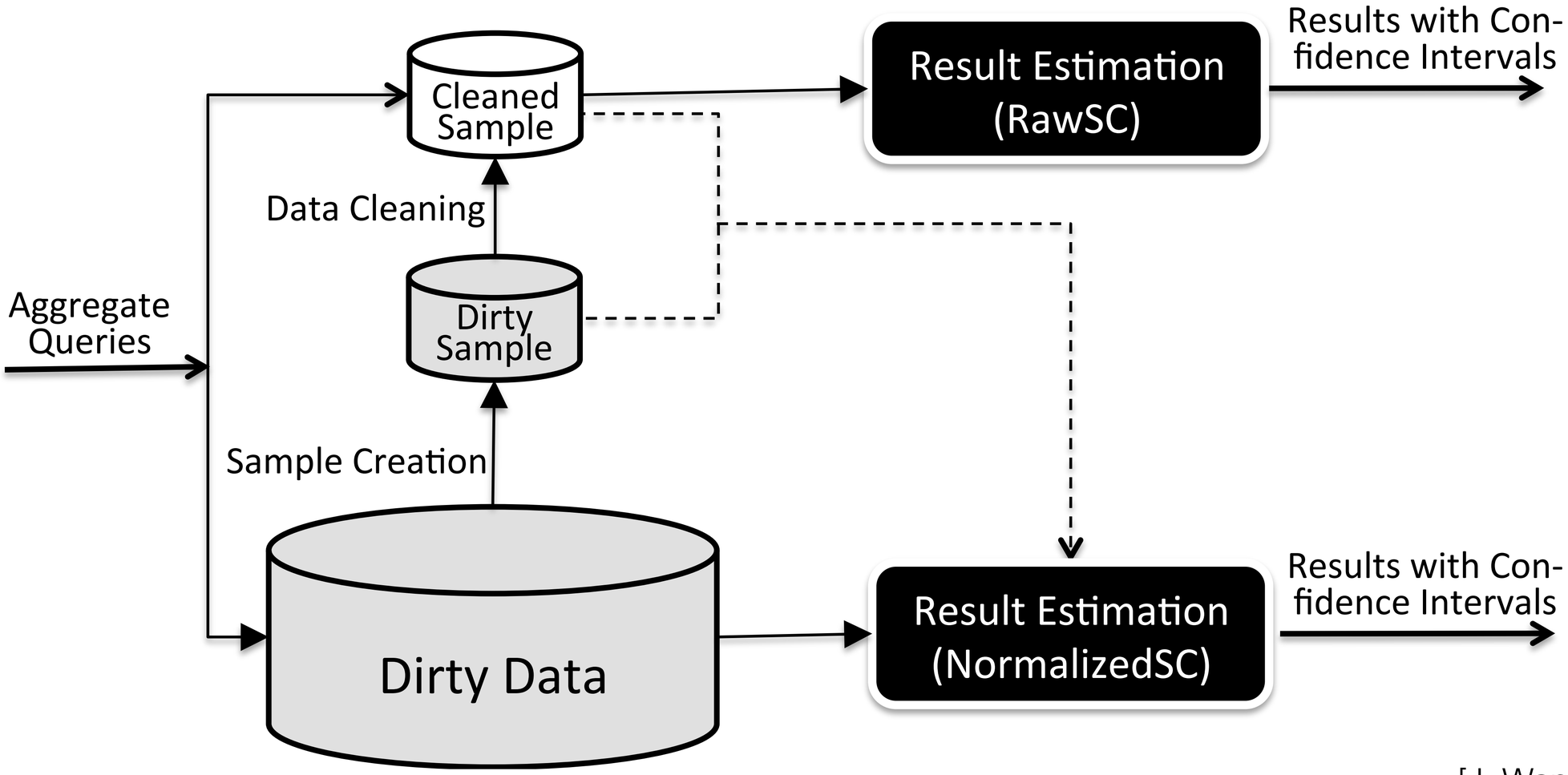
$$Q(R_{clean}) \neq Q(R) \approx est = c \cdot Q(S)$$

The error in est has two components: error due to sampling ϵ_s and error due to the difference with the cleaned relation $\epsilon_c = Q(R_{clean}) - Q(R)$:

$$| Q(R_{clean}) - est | \leq \epsilon_s + \epsilon_c$$

[S. Krishnan et al., 2015]

SampleClean Framework



[J. Wang et al., 2014]

Types of Direct Estimation Errors

- Attribute Errors:
  - value of one attribute is wrong
  - affect a single row
  - does not affect sampling
- Duplication Errors
  - same items appear multiple times
  - those items are over-represented
  - count up duplicates and divide the influence

Direct Estimation with Errors

1. Given a sample S and an aggregation function $f(\cdot)$
2. Apply $\phi_{\texttt{clean}}(\cdot)$ to each $t_i \in S$ and call the resulting set $\phi_{\texttt{clean}}(S)$
3. Calculate the mean μ_c, and the variance σ_c^2 of $\phi_{\texttt{clean}}(S)$
4. Return $\mu_c \pm \lambda\sqrt{\frac{\sigma_c^2}{K}}$

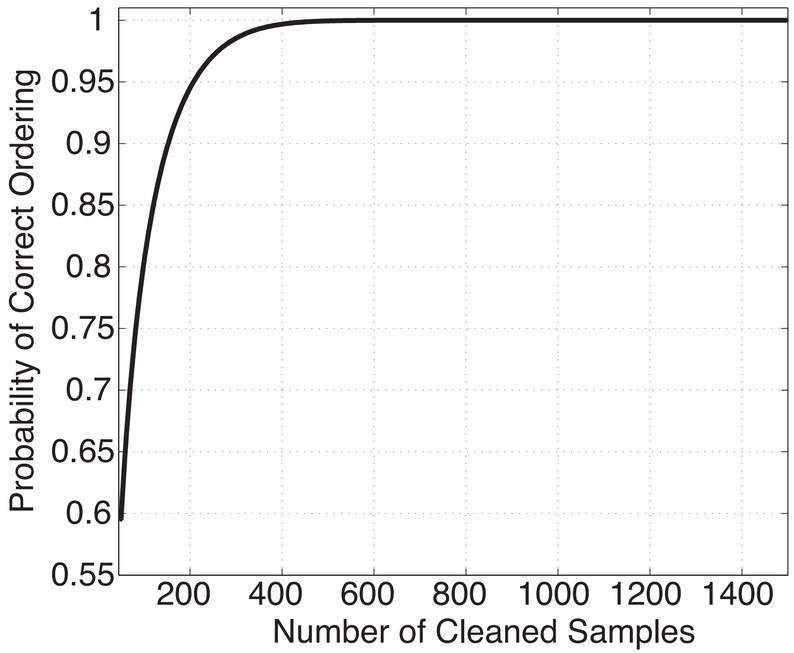
[S. Krishnan et al., 2015]

Correction with Data Errors

1. Given a sample S and an aggregation function $f(\cdot)$
2. Apply $\phi(\cdot)$ and $\phi_{\texttt{clean}}(\cdot)$ to each $r_i \in S$ and call the set of differences $Q(S)$.
3. Calculate the mean μ_q, and the variance σ_q of $Q(S)$
4. Return $(f(R) - \mu_q) \pm \lambda\sqrt{\frac{\sigma_q^2}{k}}$

[S. Krishnan et al., 2015]

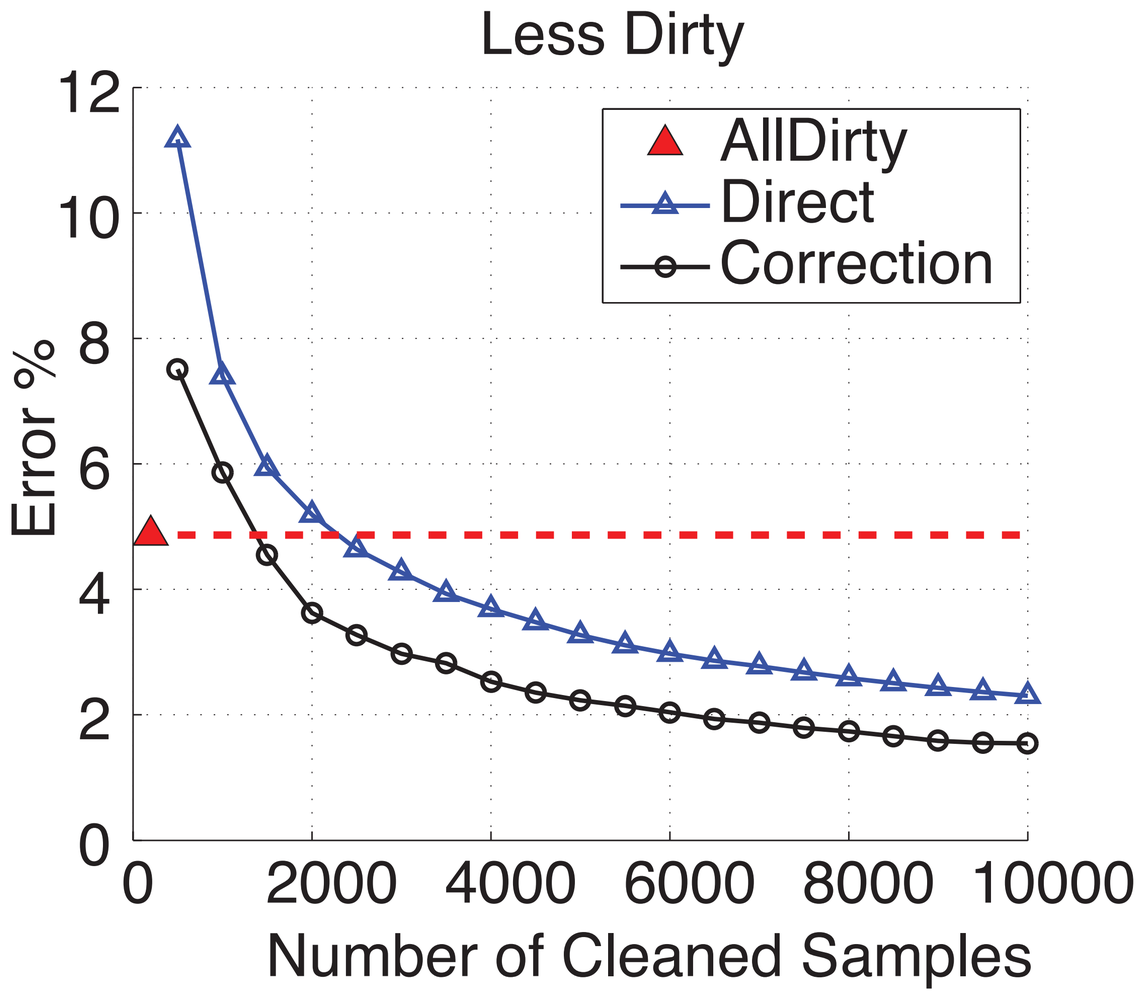
Example

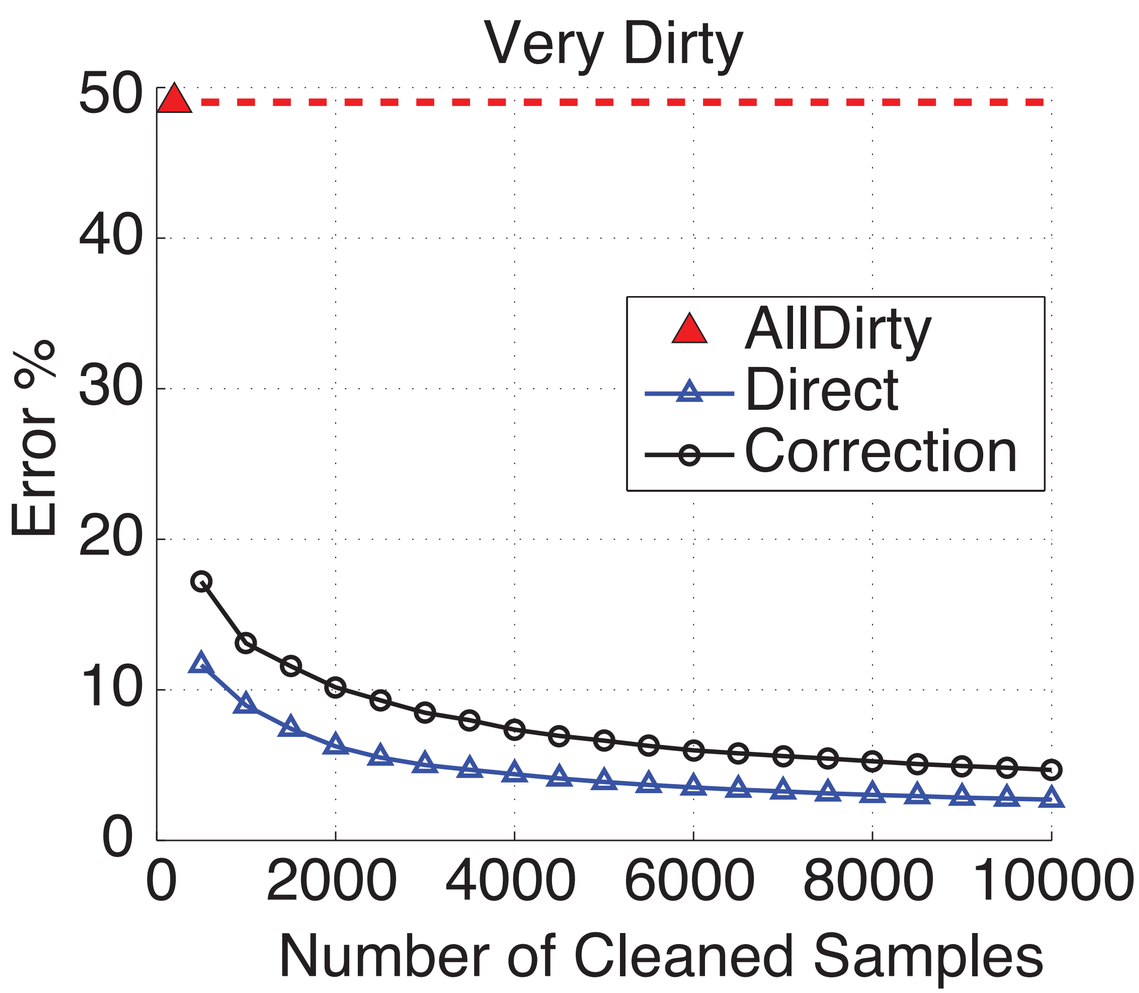


Name	Dirty	Clean	Pred %	Dup
Rakesh Agarwal	353	211	18.13%	1.28
Jeffery Ullman	460	255	05.00%	1.65
Michael Franklin	560	173	65.09%	1.13

[S. Krishnan et al., 2015]

Comparing the Two Approaches





[S. Krishnan et al., 2015]