Advanced Data Management (CSCI 680/490)

Data Cleaning

Dr. David Koop

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]

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: 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]

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: Input, Output, and Transformations

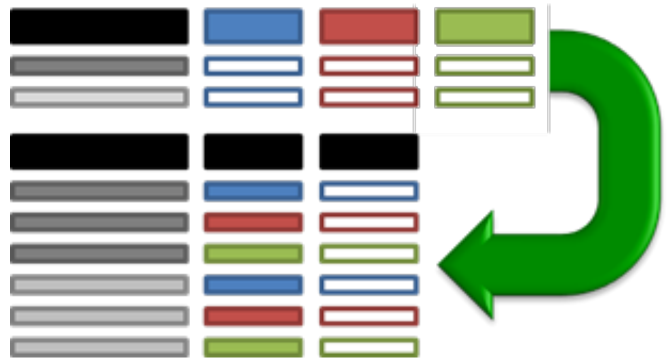
Raw Data:
- A grid of values, i.e., spreadsheets
- “Somewhat” structured - must have some regular structure or is automatically generated.

User Input:
- Sample from raw data
- Transformed view of the sample

Program to synthesize:
- A loop-free Potter’s Wheel [2] program

Transformations Targeted:

1. Layout transformation



2. String transformation

05-16-2017
05-17-2017
…

05/16/2017
05/17/2017
…

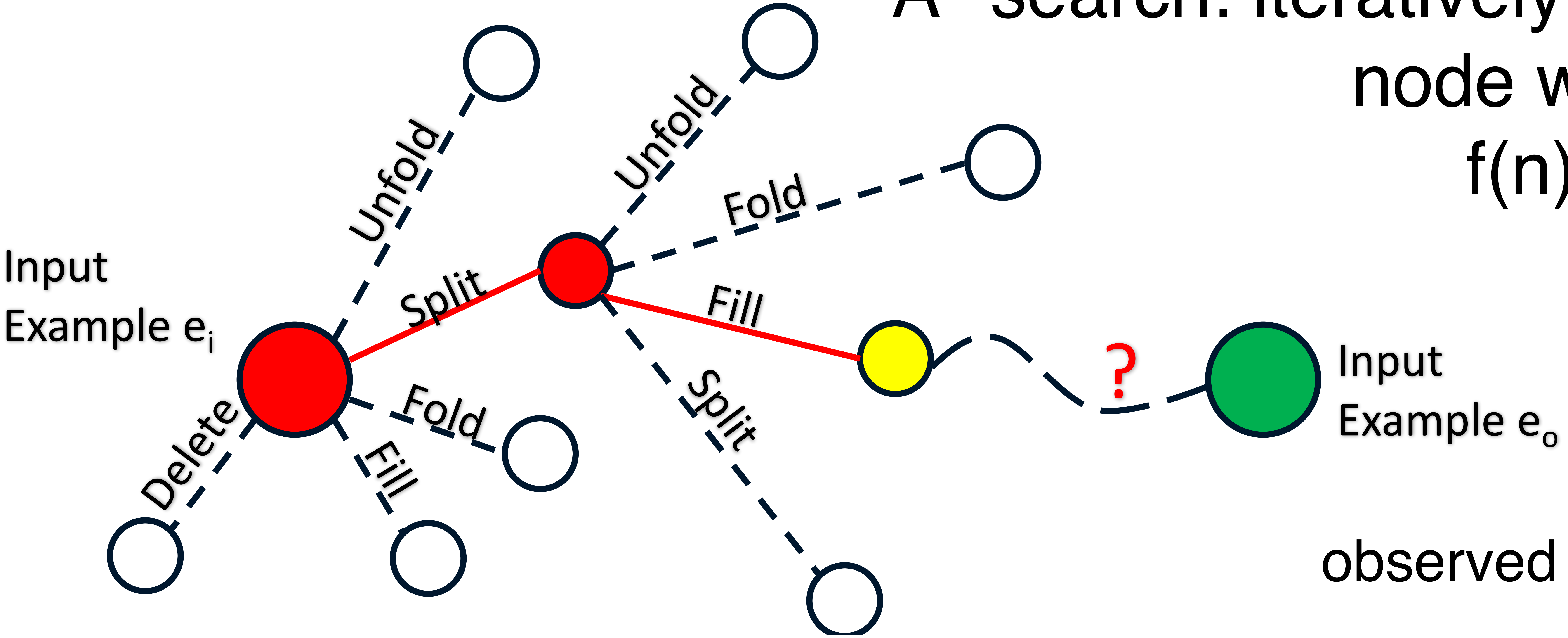
[Z. Jin et al., 2017]

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)

$$f(n) = g(n) + h(n)$$



estimated distance

observed distance

[Z. Jin et al., 2017]

AutoSuggest

- Goals:
  - Automate "Complex" Data Preparation steps
  - Focus on frame transformations (not per-cell transformations)
  - Learn from Jupyter Notebooks
  - Use **interactive** methods to help users select from top-k options
- Two Types of Predictions:
  - Single-Operator Prediction: Given two tables and an operation, decide how to best apply the operation (what are the parameters)
  - Next-Operator Prediction: Given all operations performed so far, predict the next one

[C. Yan & Y. He]

Pivot/Unpivot Prediction

- Pivot is hard to get right
  - Index
  - Header
  - Aggregation Function
  - Aggregation Columns
- Use GroupBy Prediction
- Look for NULLs and use **affiinity**
- Affinity-Maximizing Pivot Table
- Unpivot requires **compatibility**

Sector	Ticker	Company	Year	Quarter	Market Cap	Revenue
Aerospace	AJRD	AEROJET ROCKETD	2006	Q1	1442.67	472.07
Aerospace	AJRD	AEROJET ROCKETD	2006	Q2	1514.80	489.22
...	...	...	...	...	...	...
Aerospace	BA	BOEING CO	2006	Q1	343.41	210.66
...	...	...	...	...	...	...
Utilities	YORW	YORK WATER CO	2008	Q4	600.19	271.73

Sector	Ticker	Company	2006	2007	2008
Aerospace	AJRD	AEROJET ROCKETD	6218.09	6342.45	7088.62
	ATRO	ASTRONICS CORP	1050.97	1071.99	1198.11
Business Services	HHS	HARTE-HANKS INC	2473.75	2523.22	2820.07
	NCMI	NATL CINEMEDIA	856.92	874.06	976.89
Consumer Staples	YTEN	TIELD10 BIOSCI	533.13	543.79	607.77
	...	...	...	...	...
Utilities	YORW	YORK WATER CO	1902.37	1940.42	2168.70

Ticker	Company	Year	Aerospace	Business Services	...	Utilities
AJRD	AEROJET ROCKETD	2006	6218.09	NULL	...	NULL
AJRD	AEROJET ROCKETD	2007	6342.45	NULL	...	NULL
AJRD	AEROJET ROCKETD	2008	7088.62	NULL	...	NULL
ATRO	ASTRONICS CORP	2006	1050.97	NULL	...	NULL
...	...	...	...	...	...	...
HHS	HARTE-HANKS INC	2006	NULL	2473.75	...	NULL
...	...	...	...	...	...	...
YORW	YORK WATER CO	2008	NULL	NULL	...	2168.7

[C. Yan & Y. He]

AutoSuggest Results

method	full-accuracy	Rand-Index (RI)
AUTO-SUGGEST	**77%**	**0.87**
Affinity	42%	0.56
Type-Rules	19%	0.55
Min-Emptiness	46%	0.70
Balanced-Cut	14%	0.55

Table 8: Pivot: splitting index/header columns.

operator	groupby	join	concat	dropna	fillna	pivot	unpivot
percentage	33.3%	27.6%	12.2%	10.8%	9.6%	4.1%	2.4%

Table 10: Distribution of operators in data flows.

method	full accuracy	column precision	column recall	column F1
AUTO-SUGGEST	**67%**	**0.93**	**0.96**	**0.94**
Pattern-similarity	21%	0.64	0.46	0.54
Col-name-similarity	27%	0.71	0.53	0.61
Data-type	44%	0.87	0.92	0.89
Contiguous-type	46%	0.80	0.83	0.81

Table 9: Unpivot: Column prediction.

method	prec@1	prec@2	recall@1	recall@2
AUTO-SUGGEST	**0.72**	**0.79**	**0.72**	**0.85**
RNN	0.56	0.68	0.56	0.77
N-gram model	0.40	0.53	0.40	0.66
Single-Operators	0.32	0.41	0.32	0.50
Random	0.23	0.35	0.24	0.42

Table 11: Precision for next operator prediction.

[C. Yan & Y. He]

Assignment 3

- Same Salary Data
- Start with Pandas
- Hopefully Wrangler

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

Data Quality Problems

Single-Source Problems

Schema Level

(Lack of integrity constraints, poor schema design)

- Uniqueness
- Referential integrity

…

Instance Level

(Data entry errors)

- Misspellings
- Redundancy/duplicates
- Contradictory values

…

Multi-Source Problems

Schema Level

(Heterogeneous data models and schema designs)

- Naming conflicts
- Structural conflicts

…

Instance Level

(Overlapping, contradicting and inconsistent data)

- Inconsistent aggregating
- Inconsistent timing

…

[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

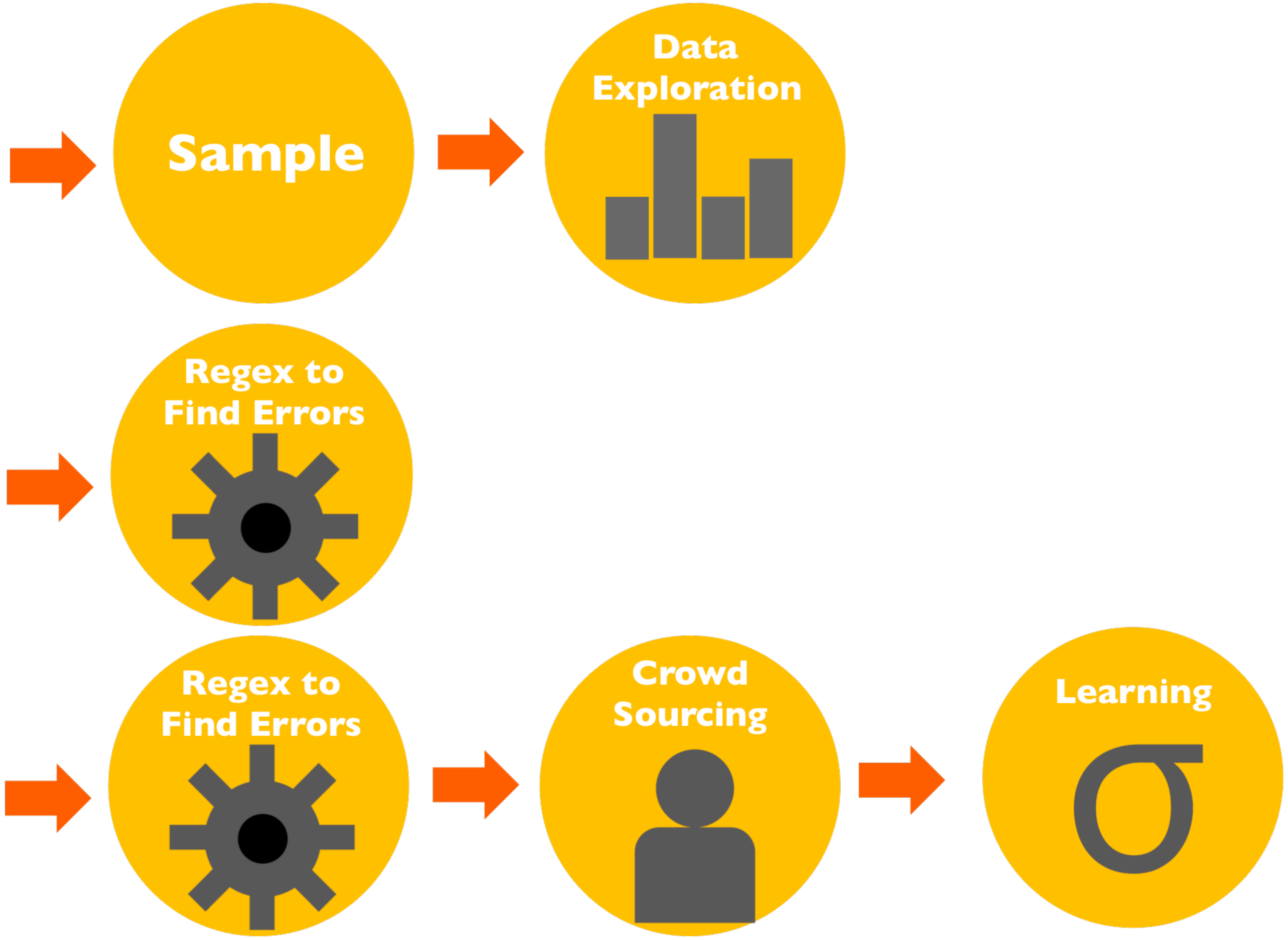
Country	UN R/P	UN R/P	World Bank	WB Gini	CIA R/P
Afghanistan	4.1;4.2		27.8	2008	
Albania	7.2	4.8	29	2012	7.2
Algeria	9.6	6.1	35.3		1995,9.6

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}.

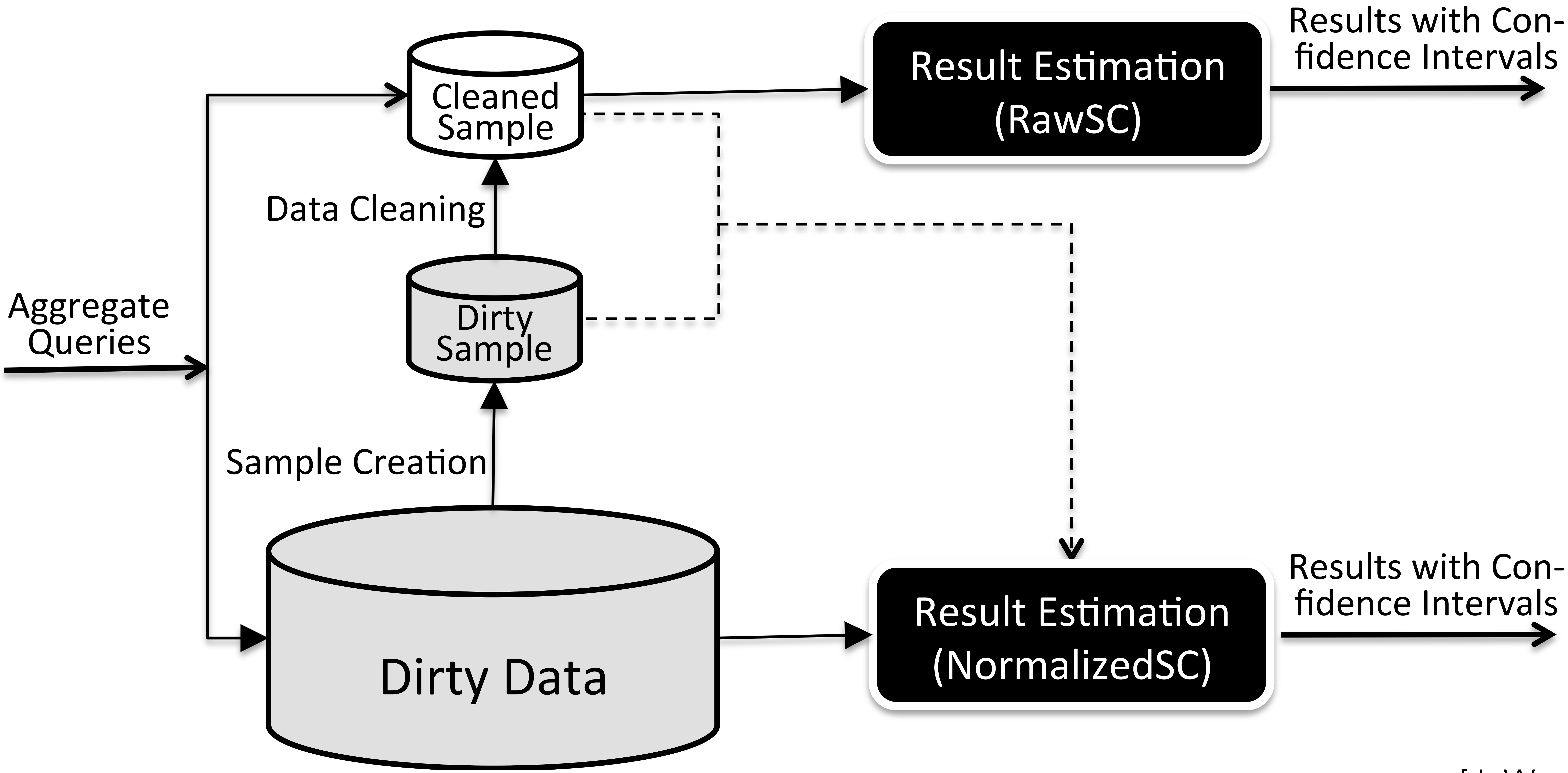
$$Q(R_{clean}) \neq Q(R) \approx est = c \cdot Q(S)$$

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

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

[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 $\mu_c \pm \lambda\sqrt{\frac{\sigma_c^2}{K}}$

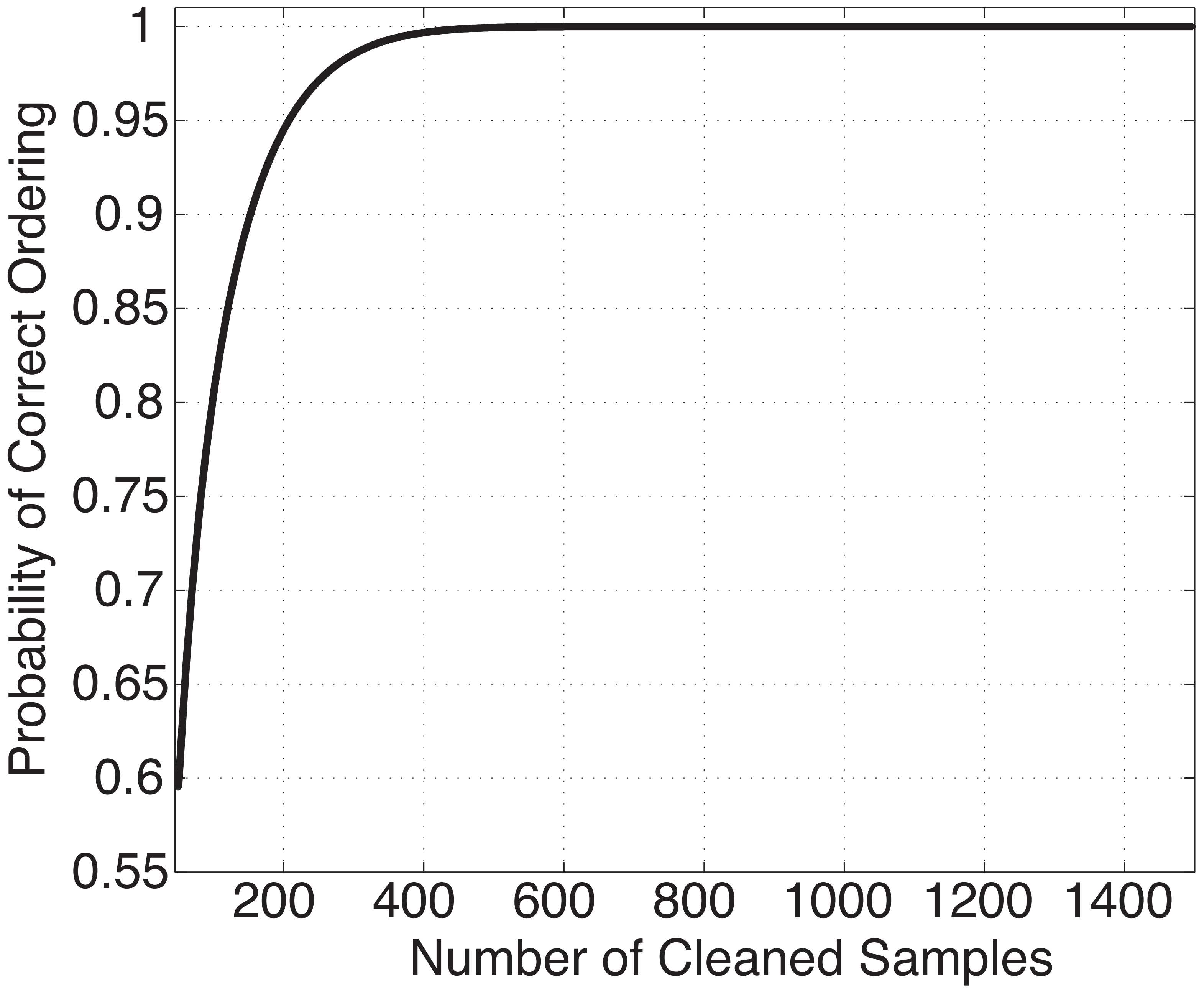
[S. Krishnan et al., 2015]

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]

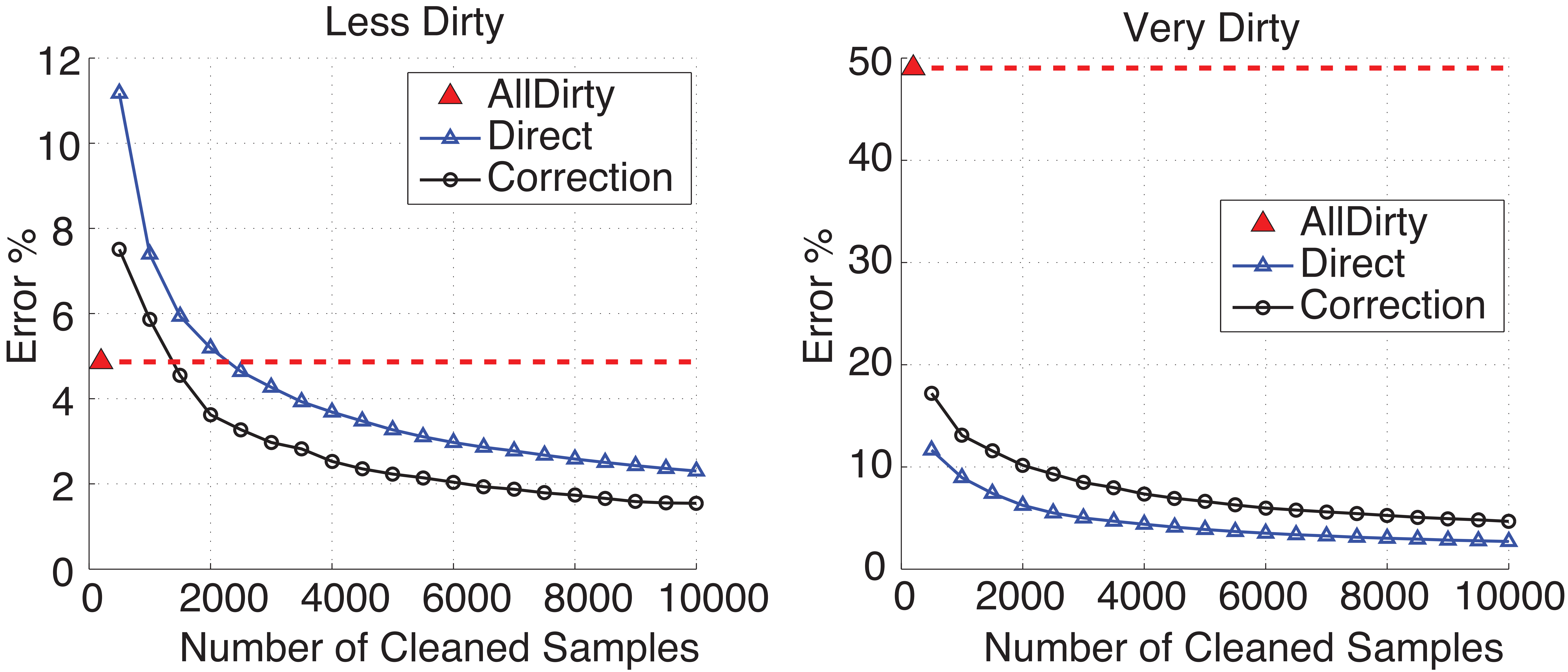
Example



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]

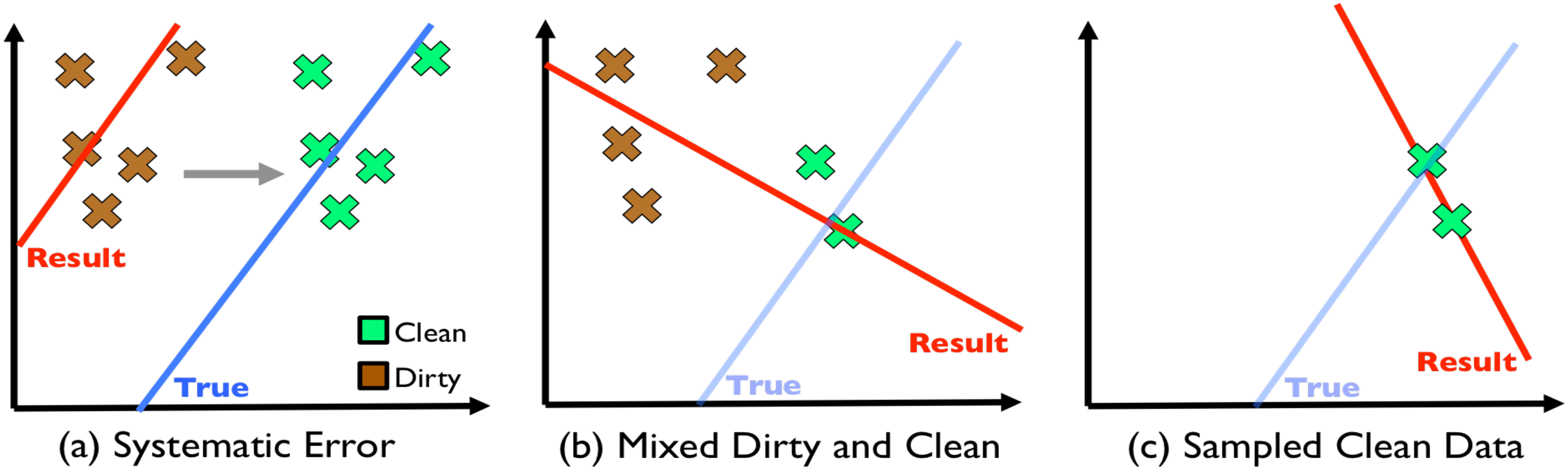
Notes

- Duplicate Problem
- Focuses on aggregate measures
- How do we actually clean the data?

[S. Krishnan et al., 2016]

Data Cleaning for Machine Learning

Problem: Simpson's Paradox



[S. Krishnan et al., 2016]

ActiveClean

- Given dirty data and a mapping from the data to a feature vector and label, we want a **reliable** estimate of the **clean** model
  - reliable = bounded estimate
- Solution: Use stochastic gradient descent (uses sampling!)

[S. Krishnan et al., 2016]

Machine Learning and Data Cleaning

- Data cleaning important for machine learning
  - Filter dirty Data
  - Make learning robust to noise (early stopping?)
- …but machine learning can also help data cleaning
  - No need for rules, just learn
  - Can include lots of features like statistical properties, integrity constraints
  - What about explainability?

HoloClean

- A holistic data cleaning framework that combines qualitative methods with quantitative methods:
  - Qualitative: use integrity constraints or external data sources
  - Quantitative: use statistics of the data
- Driven by probabilistic inference. Users only need to provide a dataset to be cleaned and describe high-level domain specific signals.
- Can scale to large real-world dirty datasets and perform automatic repairs with high accuracy

[T. Rekatsinas et al., 2017]

Example: Input Data

(A) Input Database External Information (Chicago food inspections)

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608

Conflicts due to c2

Does not obey data distribution

Conflict due to c2

(B) Functional Dependencies

$$\text{c1: DBAName} \rightarrow \text{Zip}$$
$$\text{c2: Zip} \rightarrow \text{City, State}$$
$$\text{c3: City, State, Address} \rightarrow \text{Zip}$$

[T. Rekatsinas et al., 2017]

Example: Fixing via Minimality

(B) Functional Dependencies

$$\text{c1: DBAName} \rightarrow \text{Zip}$$
$$\text{c2: Zip} \rightarrow \text{City, State}$$
$$\text{c3: City, State, Address} \rightarrow \text{Zip}$$

(A) Input Database External Information (Chicago food inspections)

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608

Conflicts due to c2

Does not obey data distribution

Conflict due to c2

(E) Repair using Minimality w.r.t FDs

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t4	**Johnnyo's**	Johnnyo's	3465 S Morgan ST	**Cicago**	IL	60608

[T. Rekatsinas et al., 2017]

Example: Fixing via External Matches

(C) Matching Dependencies

$$\text{m1: Zip} = \text{Ext\_Zip} \rightarrow \text{City} = \text{Ext\_City}$$
$$\text{m2: Zip} = \text{Ext\_Zip} \rightarrow \text{State} = \text{Ext\_State}$$
$$\text{m3: City} = \text{Ext\_City} \wedge \text{State} = \text{Ext\_State} \wedge$$
$$\wedge \text{Address} = \text{Ext\_Address} \rightarrow \text{Zip} = \text{Ext\_Zip}$$

(D) External Information (Address listings in Chicago)

Ext_Address	Ext_City	Ext_State	Ext_Zip
3465 S Morgan ST	Chicago	IL	60608
1208 N Wells ST	Chicago	IL	60610
259 E Erie ST	Chicago	IL	60611
2806 W Cermak Rd	Chicago	IL	60623

(A) Input Database External Information (Chicago food inspections)

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608

Conflicts due to c2

Does not obey data distribution

Conflict due to c2

(F) Repair using Matching Dependencies

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60608**
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60608**
t4	**Johnnyo's**	Johnnyo's	3465 S Morgan ST	**Chicago**	IL	60608

[T. Rekatsinas et al., 2017]

Example: Fixing via Statistics

(A) Input Database External Information (Chicago food inspections)

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608

Conflicts due to c2

Does not obey data distribution

Conflict due to c2

(G) Repair that leverages Quantitative Statistics

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	**60609**
t4	**John Veliotis Sr.**	Johnnyo's	3465 S Morgan ST	**Chicago**	IL	60608

[T. Rekatsinas et al., 2017]

HoloClean

Input

Dataset to be cleaned

	DBAName	Address	City	State	Zip
t1	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60609
t3	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60609
t4	Johnnyo's	3465 S Morgan ST	Cicago	IL	60608

Denial Constraints

c1: DBAName $\rightarrow$ Zip
c2: Zip $\rightarrow$ City, State
c3: City, State, Address $\rightarrow$ Zip

Matching Dependencies

m1: $\text{Zip} = \text{Ext\_Zip} \rightarrow \text{City} = \text{Ext\_City}$
m2: $\text{Zip} = \text{Ext\_Zip} \rightarrow \text{State} = \text{Ext\_State}$
m3: $\text{City} = \text{Ext\_City} \wedge \text{State} = \text{Ext\_State} \wedge \text{Address} = \text{Ext\_Address} \rightarrow \text{Zip} = \text{Ext\_Zip}$

External Information

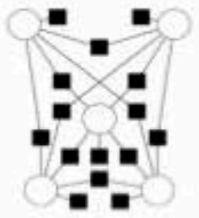
Ext_Address	Ext_City	Ext_State	Ext_Zip
3465 S Morgan ST	Chicago	IL	60608
1208 N Wells ST	Chicago	IL	60610
259 E Erie ST	Chicago	IL	60611
2806 W Cermak Rd	Chicago	IL	60623

The HoloClean Framework

1. Error detection module

2. Automatic compilation to a probabilistic graphical model

3. Repair via statistical learning and infernece



Output

Proposed Cleaned Dataset

	DBAName	Address	City	State	Zip
t1	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	60608
t2	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	**60608**
t3	John Veliotis Sr.	3465 S Morgan ST	Chicago	IL	**60608**
t4	**John Veliotis Sr.**	3465 S Morgan ST	**Chicago**	IL	60608

Marginal Distribution of Cell Assignments

Cell	Possible Values	Probability
t2.Zip	60608	0.84
	60609	0.16
t4.City	Chicago	0.95
	Cicago	0.05
t4.DBAName	John Veliotis Sr.	0.99
	Johnnyo's	0.01

[T. Rekatsinas et al., 2017]

Data Cleaning in pandas

Handling Missing Data

- Filtering out missing data:
  - Can choose rows or columns
- Filling in missing data:
  - with a default value
  - with an interpolated value
- In pandas:

Argument	Description
`dropna`	Filter axis labels based on whether values for each label have missing data, with varying thresholds for how much missing data to tolerate.
`fillna`	Fill in missing data with some value or using an interpolation method such as `'ffill'` or `'bfill'`.
`isnull`	Return boolean values indicating which values are missing/NA.
`notnull`	Negation of `isnull`.

[W. McKinney, Python for Data Analysis]

Filling in missing data

- fillna arguments:

Argument	Description
`value`	Scalar value or dict-like object to use to fill missing values
`method`	Interpolation; by default `'ffill'` if function called with no other arguments
`axis`	Axis to fill on; default `axis=0`
`inplace`	Modify the calling object without producing a copy
`limit`	For forward and backward filling, maximum number of consecutive periods to fill

[W. McKinney, Python for Data Analysis]

Filtering and Cleaning Data

- Find duplicates
  - `duplicated`: returns boolean Series indicating whether row is a duplicate—first instance is **not marked** as a duplicate
- Remove duplicates:
  - `drop_duplicates`: drops all rows where `duplicated` is `True`
  - `keep`: which value to keep (first or last)
- Can pass specific columns to check for duplicates, e.g. check only key column

Changing Data

- Convert strings to upper/lower case
- Convert Fahrenheit temperatures to Celsius
- Create a new column based on another column

```
In [56]: lowercased
Out[56]:
0          bacon
1    pulled pork
2          bacon
3       pastrami
4    corned beef
5          bacon
6       pastrami
7      honey ham
8       nova lox
Name: food, dtype: object
```

```
meat_to_animal = {
  'bacon': 'pig',
  'pulled pork': 'pig',
  'pastrami': 'cow',
  'corned beef': 'cow',
  'honey ham': 'pig',
  'nova lox': 'salmon'
}
```

```
In [57]: data['animal'] = lowercased.map(meat_to_animal)

In [58]: data
Out[58]:
          food  ounces  animal
0        bacon     4.0     pig
1  pulled pork     3.0     pig
2        bacon    12.0     pig
3     Pastrami     6.0     cow
4  corned beef     7.5     cow
5        Bacon     8.0     pig
6     pastrami     3.0     cow
7    honey ham     5.0     pig
8     nova lox     6.0  salmon
```

[W. McKinney, Python for Data Analysis]

Replacing Values

- `fillna` is a special case
- What if `-999` in our dataset was identified as a missing value?

```
In [61]: data
Out[61]:
0       1.0
1    -999.0
2       2.0
3    -999.0
4   -1000.0
5       3.0
dtype: float64
```

```
In [62]: data.replace(-999, np.nan)
Out[62]:
0       1.0
1       NaN
2       2.0
3       NaN
4   -1000.0
5       3.0
dtype: float64
```

- Can pass list of values or dictionary to change different values

Clamping Values

- Values above or below a specified thresholds are set to a max/min value

```
In [93]: data.describe()
Out[93]:
                 0            1            2            3
count  1000.000000  1000.000000  1000.000000  1000.000000
mean      0.049091     0.026112    -0.002544    -0.051827
std       0.996947     1.007458     0.995232     0.998311
min      -3.645860    -3.184377    -3.745356    -3.428254
25%      -0.599807    -0.612162    -0.687373    -0.747478
50%       0.047101    -0.013609    -0.022158    -0.088274
75%       0.756646     0.695298     0.699046     0.623331
max       2.653656     3.525865     2.735527     3.366626
```

```
In [97]: data[np.abs(data) > 3] = np.sign(data) * 3

In [98]: data.describe()
Out[98]:
                 0            1            2            3
count  1000.000000  1000.000000  1000.000000  1000.000000
mean      0.050286     0.025567    -0.001399    -0.051765
std       0.992920     1.004214     0.991414     0.995761
min      -3.000000    -3.000000    -3.000000    -3.000000
25%      -0.599807    -0.612162    -0.687373    -0.747478
50%       0.047101    -0.013609    -0.022158    -0.088274
75%       0.756646     0.695298     0.699046     0.623331
max       2.653656     3.000000     2.735527     3.000000
```