DS-GA 3001.009: Responsible Data Science

Data Profiling continued

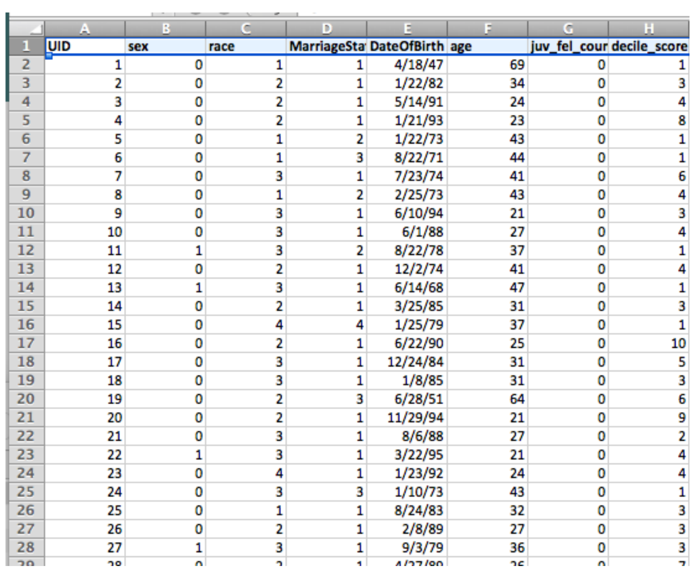
Prof. Julia Stoyanovich
Center for Data Science
Computer Science and Engineering at Tandon

@stoyanoj

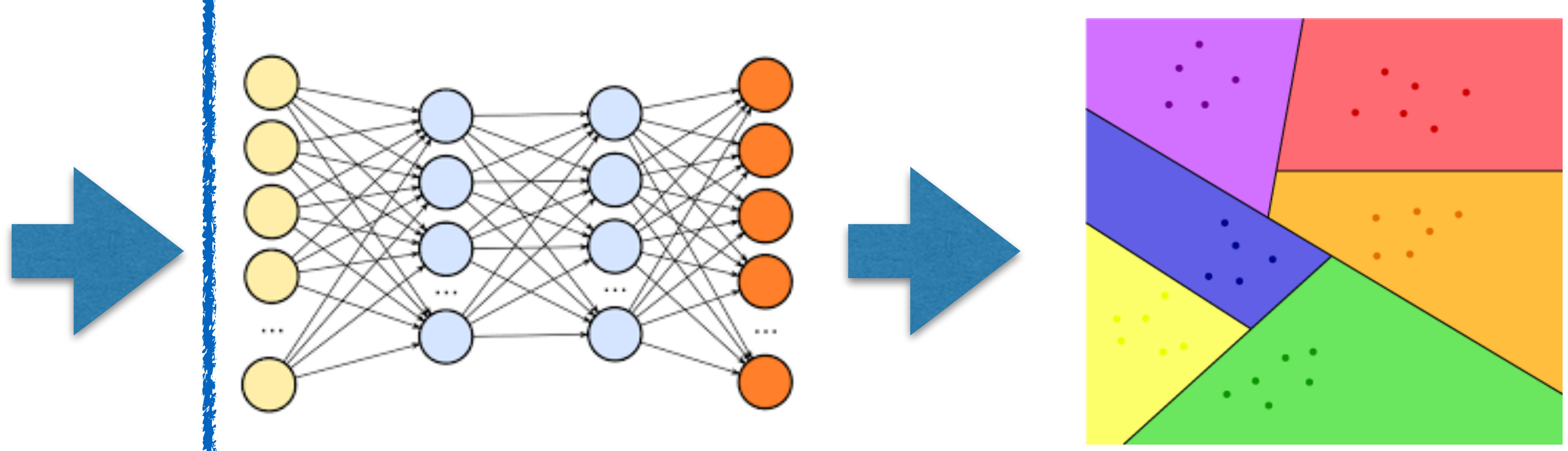
http://stoyanovich.org/
https://dataresponsibly.github.io/

Responsible data science

- Be **transparent** and **accountable**
- Achieve **equitable** resource distribution
- Be cognizant of the **rights** and **preferences** of individuals



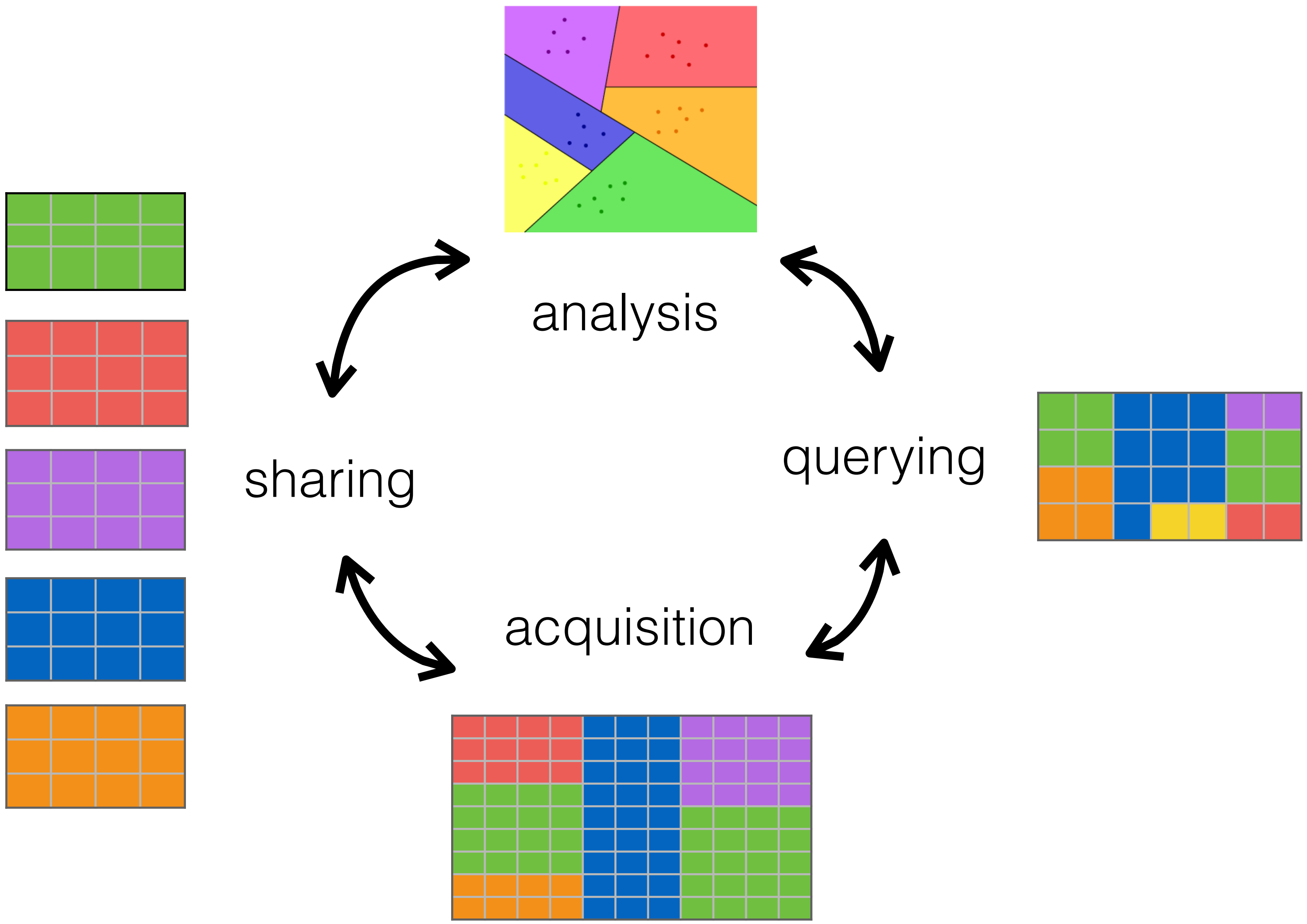
	A	B	C	D	E	F	G	H
1	UID	sex	race	MarriageSta	DateOfBirth	age	juv_fel_cour	decile_score
2	1	0	1	1	4/18/47	69	0	1
3	2	0	2	1	1/22/82	34	0	3
4	3	0	2	1	5/14/91	24	0	4
5	4	0	2	1	1/21/93	23	0	8
6	5	0	1	2	1/22/73	43	0	1
7	6	0	1	3	8/22/71	44	0	1
8	7	0	3	1	7/23/74	41	0	6
9	8	0	1	2	2/25/73	43	0	4
10	9	0	3	1	6/10/94	21	0	3
11	10	0	3	1	6/1/88	27	0	4
12	11	1	3	2	8/22/78	37	0	1
13	12	0	2	1	12/2/74	41	0	4
14	13	1	3	1	6/14/68	47	0	1
15	14	0	2	1	3/25/85	31	0	3
16	15	0	4	4	1/25/79	37	0	1
17	16	0	2	1	6/22/90	25	0	10
18	17	0	3	1	12/24/84	31	0	5
19	18	0	3	1	1/8/85	31	0	3
20	19	0	2	3	6/28/51	64	0	6
21	20	0	2	1	11/29/94	21	0	9
22	21	0	3	1	8/6/88	27	0	2
23	22	1	3	1	3/22/95	21	0	4
24	23	0	4	1	1/23/92	24	0	4
25	24	0	3	3	1/10/73	43	0	1
26	25	0	1	1	8/24/83	32	0	3
27	26	0	2	1	2/8/89	27	0	3
28	27	1	3	1	9/3/79	36	0	3



done?

but where does the data come from?

A holistic view of the lifecycle



Understand your data!

CRA
Computing Research Association

COMPUTING COMMUNITY CONSORTIUM

"Given the heterogeneity of the flood of data, it is **not enough merely to record it and throw it into a repository**. Consider, for example, data from a range of scientific experiments. If we just have a bunch of data sets in a repository, it is **unlikely anyone will ever be able to find, let alone reuse**, any of this data. With adequate **metadata**, there is some hope, but even so, challenges will remain due to differences in experimental details and in data record structure."

https://cra.org/ccc/wp-content/uploads/sites/2/2015/05/bigdatawhitepaper.pdf

Understand your data!

2.2 Big data

In the analog age, most of the data that were used for social research was created for the purpose of doing research. In the digital age, however, a huge amount of **data is being created by companies and governments for purposes other than research**, such as providing services, generating profit, and administering laws. Creative people, however, have realized that you can ***repurpose*** this corporate and government data for research.

https://www.bitbybitbook.com/en/1st-ed/observing-behavior/data/

Understand your data!

2.2 Big data

… from the perspective of researchers, big data sources are “found,” they don’t just fall from the sky. Instead, data sources that are “found” by researchers are **designed by someone for some purpose**. Because “found” data are designed by someone, I always recommend that you **try to understand as much as possible about the people and processes that created your data**.

https://www.bitbybitbook.com/en/1st-ed/observing-behavior/data/

Understand your data!

Need **metadata** to:

- enable data **re-use** (have to be able to find it!)
- determine **fitness for use** of a dataset in a task
- help establish **trust** in the data analysis process and its outcomes

Data is considered to be of high quality if it's "**fit for intended uses** in operations, decision making and planning"

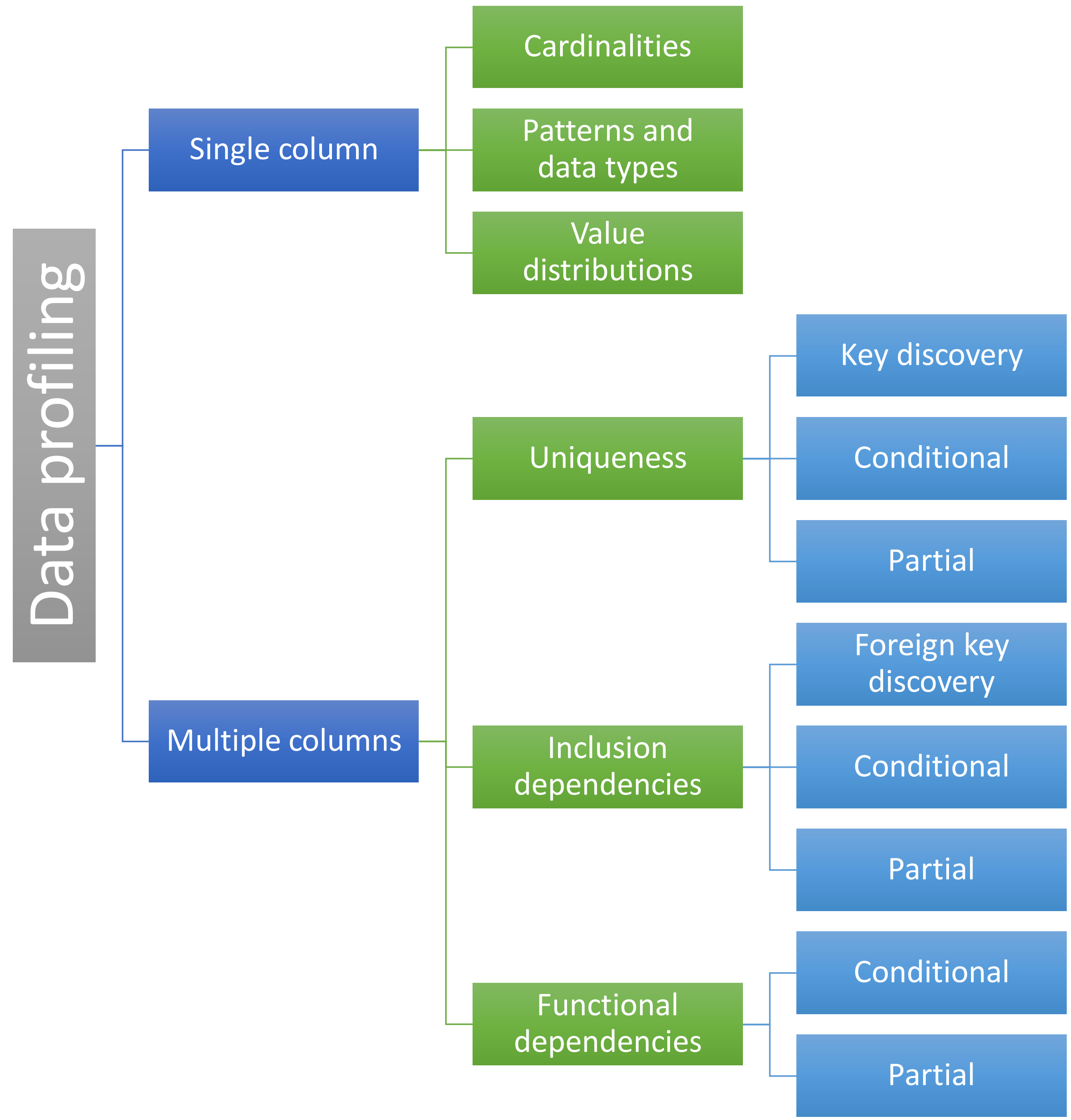
[Thomas C. Redman, "Data Driven: Profiting from Your Most Important Business Asset." 2013]

A classification of data profiling tasks

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

	A	B	C	D	E	F	G	H
1	UID	sex	race	MarriageSta	DateOfBirth	age	juv_fel_cour	decile_score
2	1	0	1	1	4/18/47	69	0	1
3	2	0	2	1	1/22/82	34	0	3
4	3	0	2	1	5/14/91	24	0	4
5	4	0	2	1	1/21/93	23	0	8
6	5	0	1	2	1/22/73	43	0	1
7	6	0	1	3	8/22/71	44	0	1
8	7	0	3	1	7/23/74	41	0	6
9	8	0	1	2	2/25/73	43	0	4
10	9	0	3	1	6/10/94	21	0	3
11	10	0	3	1	6/1/88	27	0	4
12	11	1	3	2	8/22/78	37	0	1
13	12	0	2	1	12/2/74	41	0	4
14	13	1	3	1	6/14/68	47	0	1
15	14	0	2	1	3/25/85	31	0	3
16	15	0	4	4	1/25/79	37	0	1
17	16	0	2	1	6/22/90	25	0	10
18	17	0	3	1	12/24/84	31	0	5
19	18	0	3	1	1/8/85	31	0	3
20	19	0	2	3	6/28/51	64	0	6
21	20	0	2	1	11/29/94	21	0	9
22	21	0	3	1	8/6/88	27	0	2
23	22	1	3	1	3/22/95	21	0	4
24	23	0	4	1	1/23/92	24	0	4
25	24	0	3	3	1/10/73	43	0	1
26	25	0	1	1	8/24/83	32	0	3
27	26	0	2	1	2/8/89	27	0	3
28	27	1	3	1	9/3/79	36	0	3

relational data (here: just one table)



An alternative classification

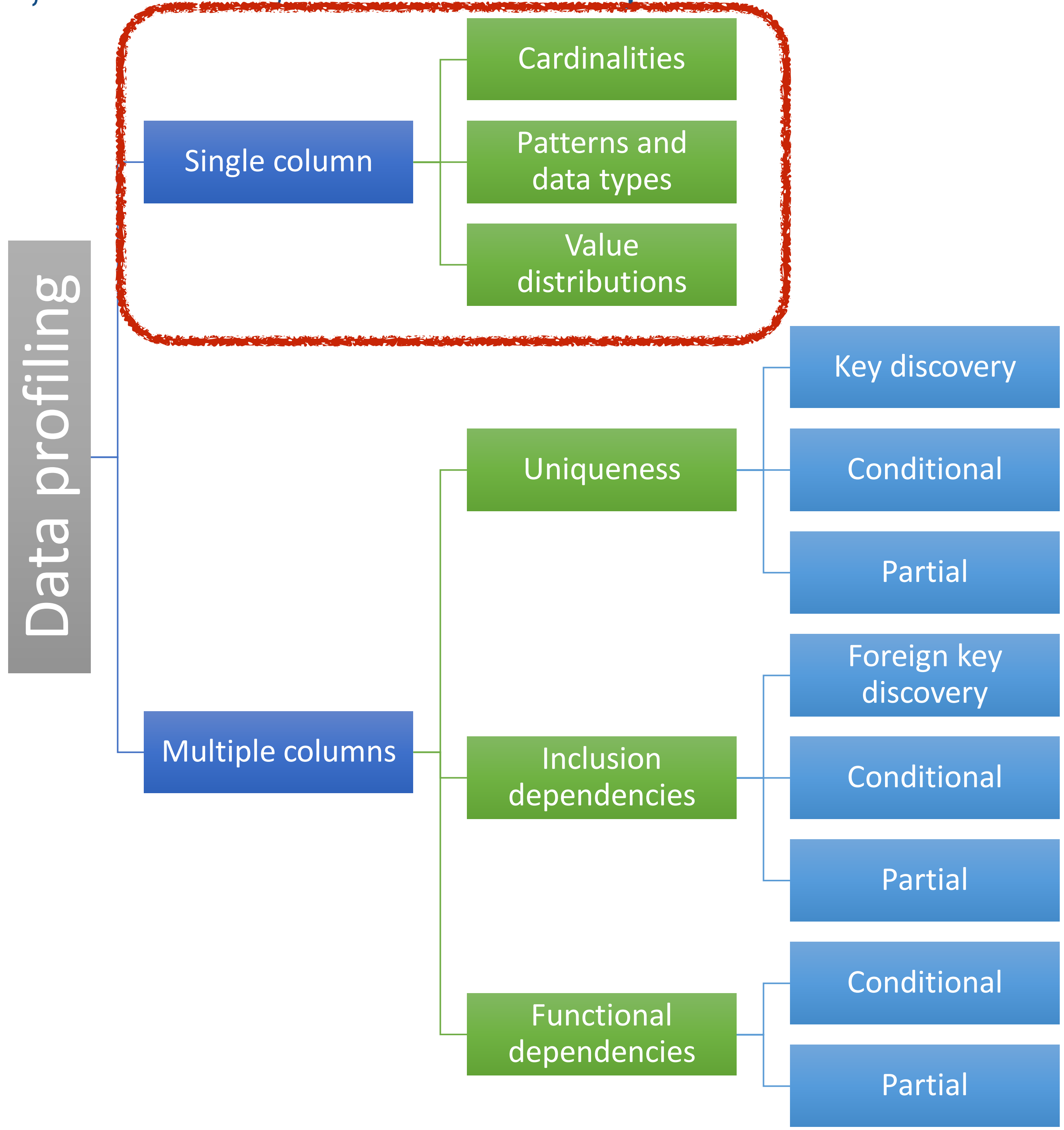
- To help understand the **statistics**, we look at value ranges, data types, value distributions per column or across columns, etc
- To help understand the **structure** - the (business) rules that generated the data - we look at unique columns / column combinations, dependencies between columns, etc - **reverse-engineer the relational schema** of the data we have
- We need both statistics and structure, they are mutually-reinforcing, and help us understand the **semantics** of the data - it's meaning

A classification of data profiling tasks

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

	A	B	C	D	E	F	G	H
1	UID	sex	race	MarriageSta	DateOfBirth	age	juv_fel_cour	decile_score
2	1	0	1	1	4/18/47	69	0	1
3	2	0	2	1	1/22/82	34	0	3
4	3	0	2	1	5/14/91	24	0	4
5	4	0	2	1	1/21/93	23	0	8
6	5	0	1	2	1/22/73	43	0	1
7	6	0	1	3	8/22/71	44	0	1
8	7	0	3	1	7/23/74	41	0	6
9	8	0	1	2	2/25/73	43	0	4
10	9	0	3	1	6/10/94	21	0	3
11	10	0	3	1	6/1/88	27	0	4
12	11	1	3	2	8/22/78	37	0	1
13	12	0	2	1	12/2/74	41	0	4
14	13	1	3	1	6/14/68	47	0	1
15	14	0	2	1	3/25/85	31	0	3
16	15	0	4	4	1/25/79	37	0	1
17	16	0	2	1	6/22/90	25	0	10
18	17	0	3	1	12/24/84	31	0	5
19	18	0	3	1	1/8/85	31	0	3
20	19	0	2	3	6/28/51	64	0	6
21	20	0	2	1	11/29/94	21	0	9
22	21	0	3	1	8/6/88	27	0	2
23	22	1	3	1	3/22/95	21	0	4
24	23	0	4	1	1/23/92	24	0	4
25	24	0	3	3	1/10/73	43	0	1
26	25	0	1	1	8/24/83	32	0	3
27	26	0	2	1	2/8/89	27	0	3
28	27	1	3	1	9/3/79	36	0	3

relational data (here: just one table)



NYU

Single column: cardinalities, data types

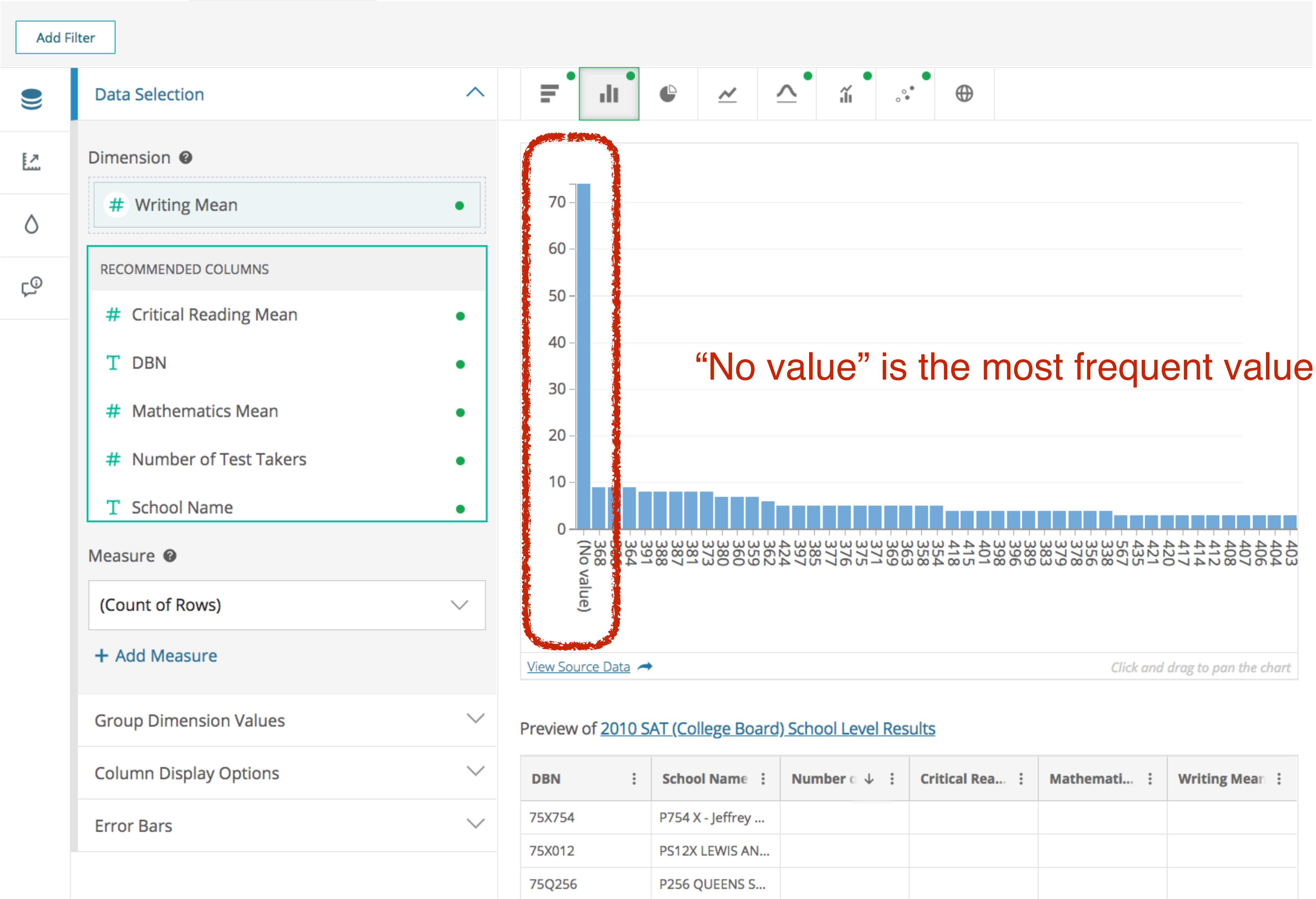
[Abedjan, Golab, Naumann; *SIGMOD 2017*]

- cardinality of relation **R** - number of rows
- domain cardinality of a column **R.a** - number of **distinct** values
- attribute value **length**: min, max, average, median
- **basic data type**: string, numeric, date, time, ….
- number of percentage of **null** values of a given attribute
- regular expressions
- semantic domain: SSN, phone number
- ….

2010 SAT (College Board) School Level Results

Education

NYC OpenData



50 shades of null

- **Unknown** - some value definitely belongs here, but I don't know what it is (e.g., unknown birthdate)
- **Inapplicable** - no value makes sense here (e.g., if marital status = single then spouse name should not have a value)
- **Unintentionally omitted** - values is left unspecified unintentionally, by mistake
- **Optional** - a value may legitimately be left unspecified (e.g., middle name)
- **Intentionally withheld** (e.g., an unlisted phone number)
- …..

(this selection is mine, see reference below for a slightly different list)

https://www.vertabelo.com/blog/technical-articles/50-shades-of-null-or-how-a-billion-dollar-mistake-has-been-stalking-a-whole-industry-for-decades

50 shades of null… and it gets worse!

- **Hidden missing values** -
  - 99999 for zip code, Alabama for state
  - need data cleaning….
- lots of houses in Philadelphia, PA were built in 1934 (or 1936?) - not really!

how do we detect hidden missing values?

Single column: cardinalities, data types

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

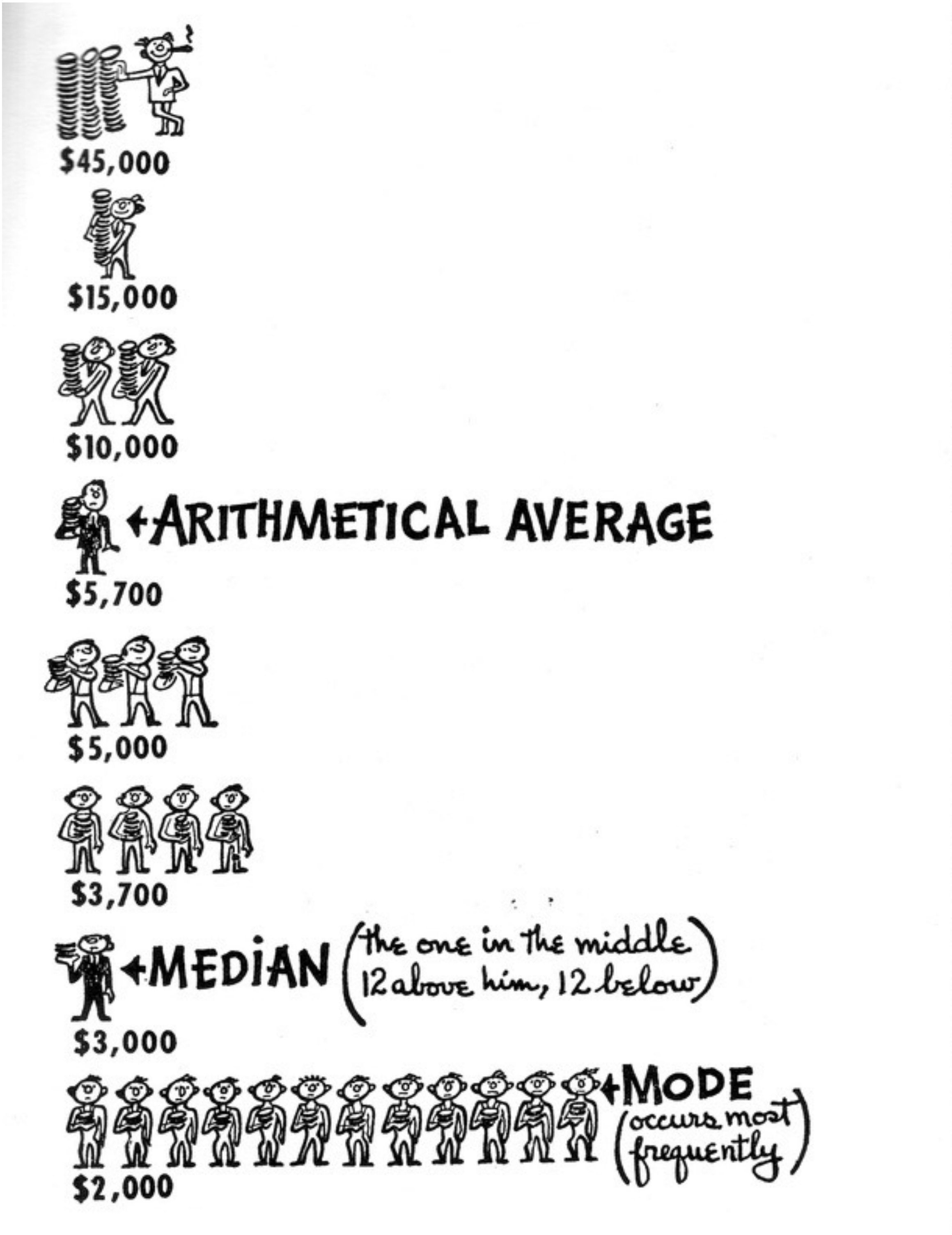
- cardinality of relation **R** - number of rows
- domain cardinality of a column **R.a** - number of **distinct** values
- attribute value **length**: min, max, average, median
- **basic data type**: string, numeric, date, time, ….
- number of percentage of **null** values of a given attribute
- **regular expressions**
- semantic domain: SSN, phone number
- ….

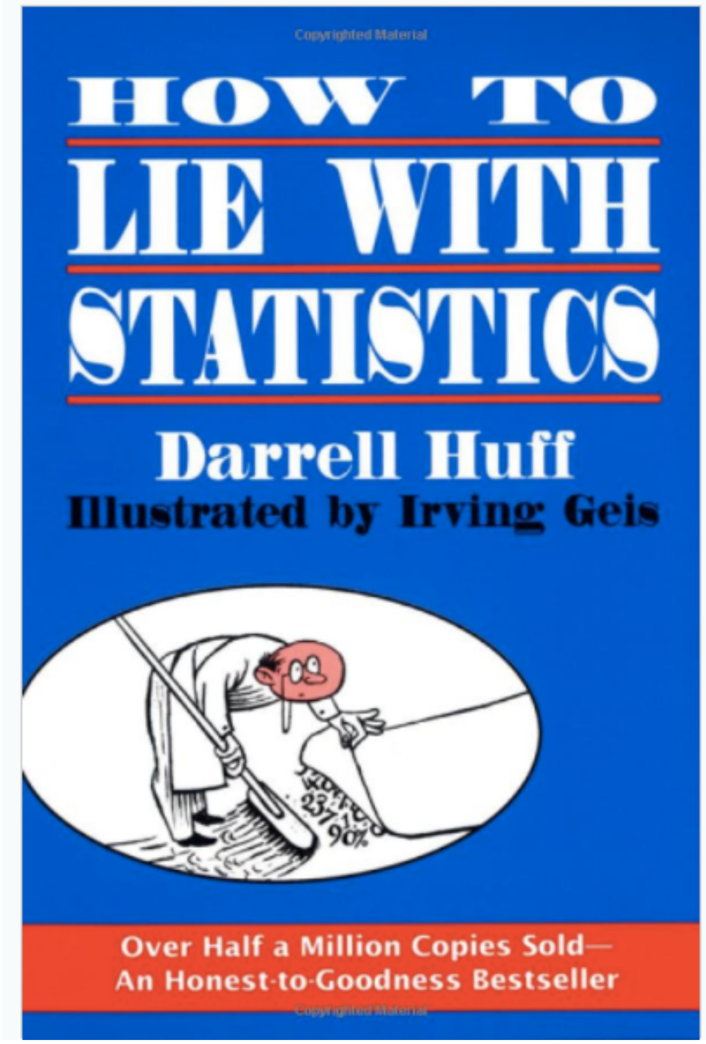
Single column: basic stats, distributions

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

- min, max, **average**, median value of **R.a**
- **histogram**
  - equi-width - (approximately) the same number of distinct values in each bucket (e.g., age broken down into 5-year windows)
  - equi-depth (approximately) the same number of tuples in each bucket
  - biased histograms use different granularities for different parts of the value range to provide better accuracy
- quartiles - three points that divide the numeric values into four equal groups - a kind of an equi-depth histogram
- **first digit** - distribution of first digit in numeric values, to check Benford law
- …

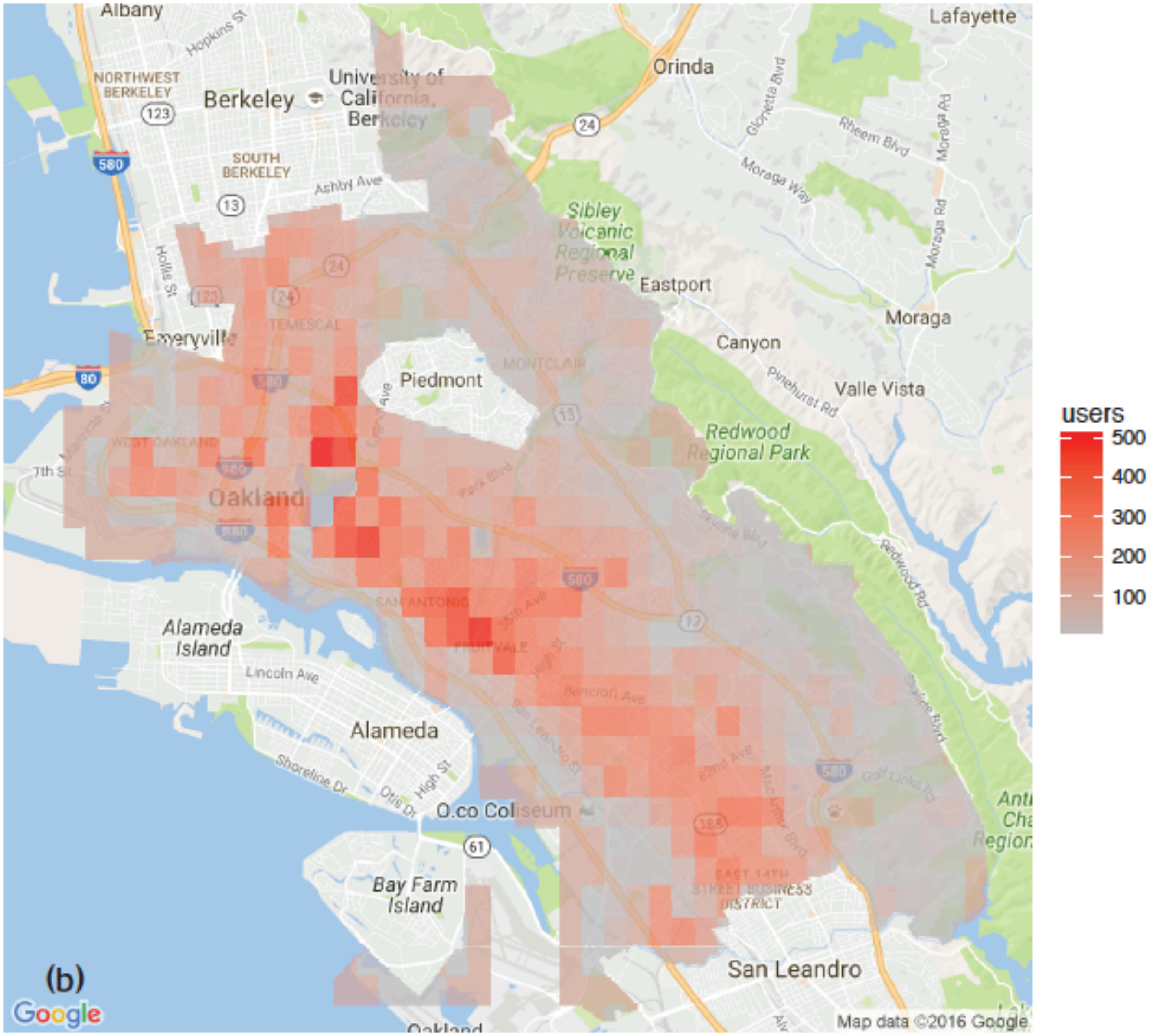
The well-chosen average



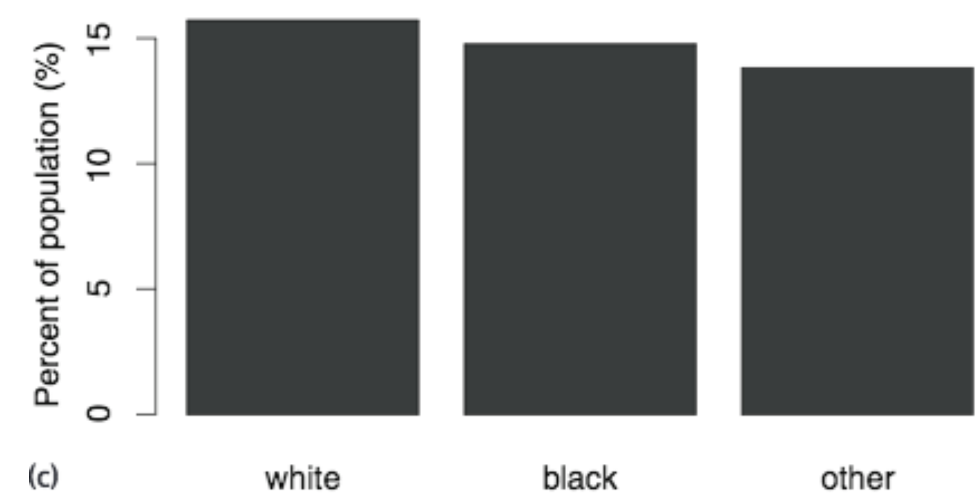


Is my data biased? (histograms + geo)

[Lum, Isaac; *Significance, 2016*]



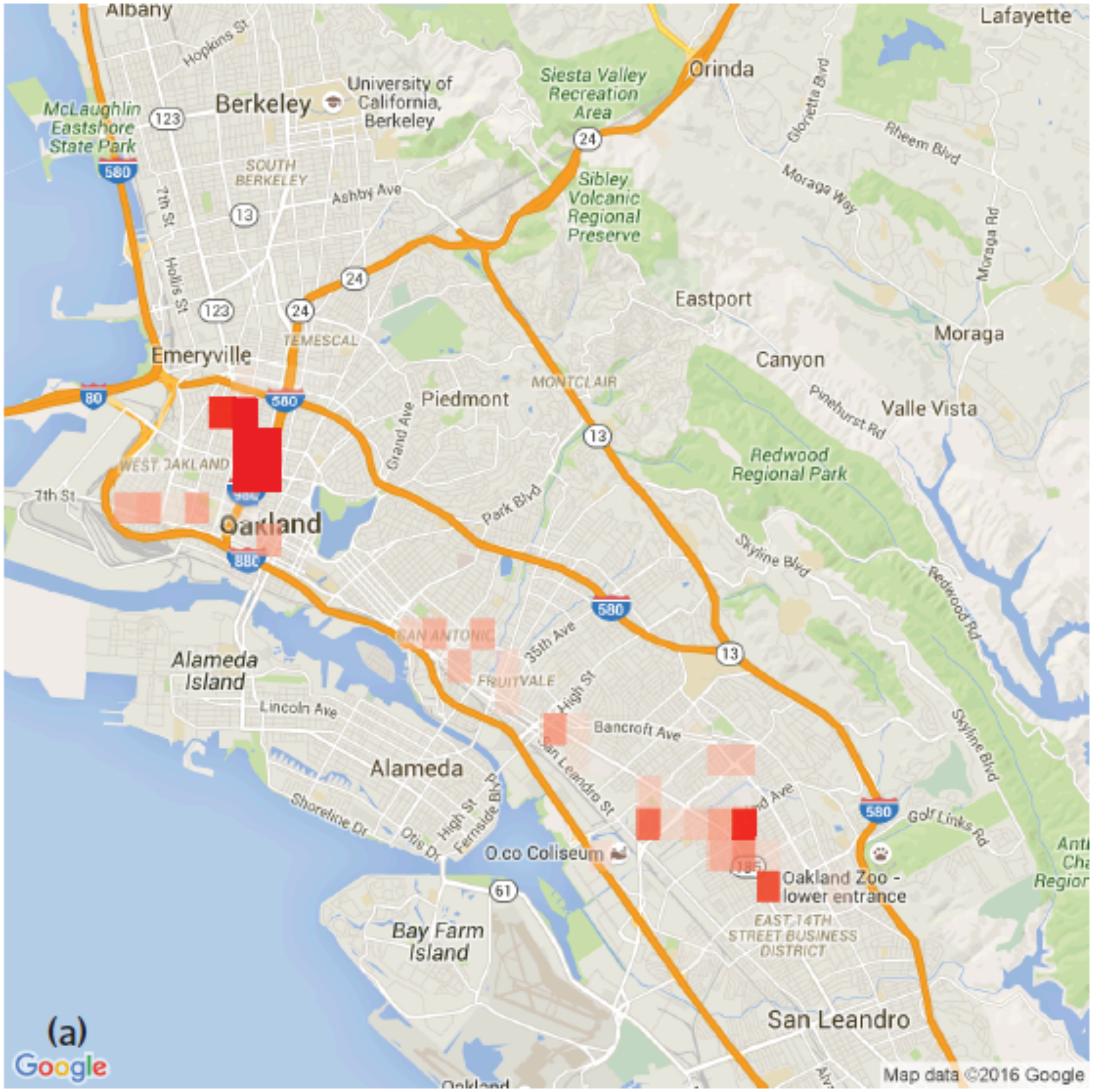
Estimated number of drug users, based on 2011 National Survey on Drug Use and Health, in Oakland, CA



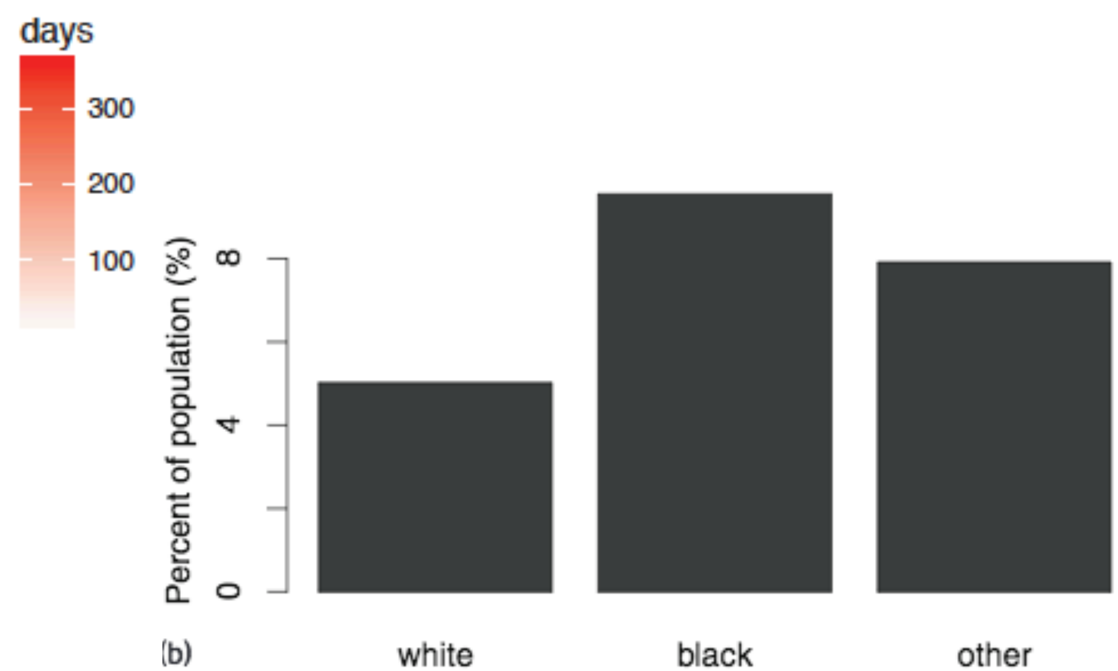
Estimated drug use by race

Is my data biased? (histograms + geo)

[Lum, Isaac; *Significance, 2016*]



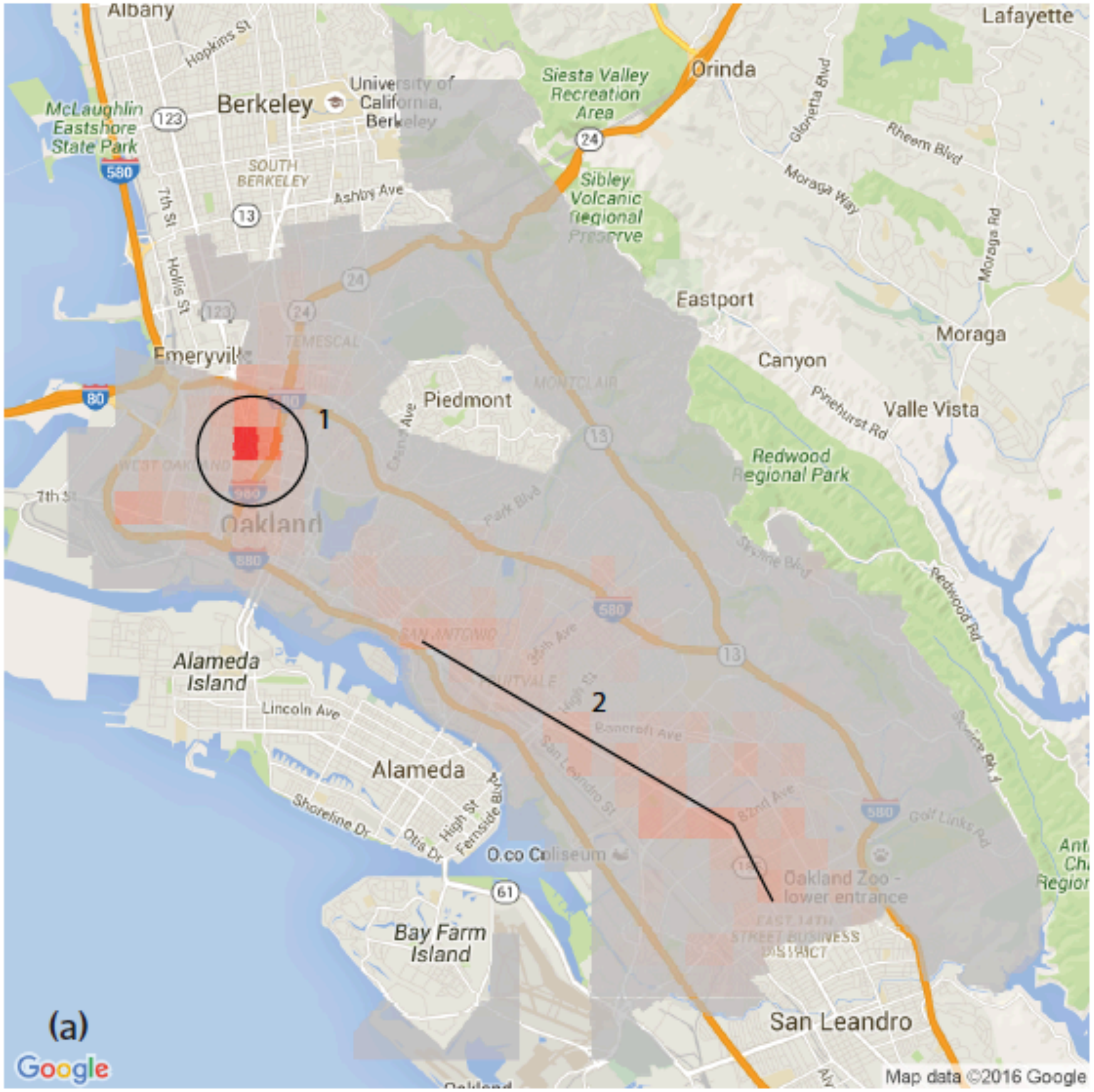
Number of days with targeted policing for drug crimes in areas flagged by PredPol analysis of Oakland, CA, police data for 2011



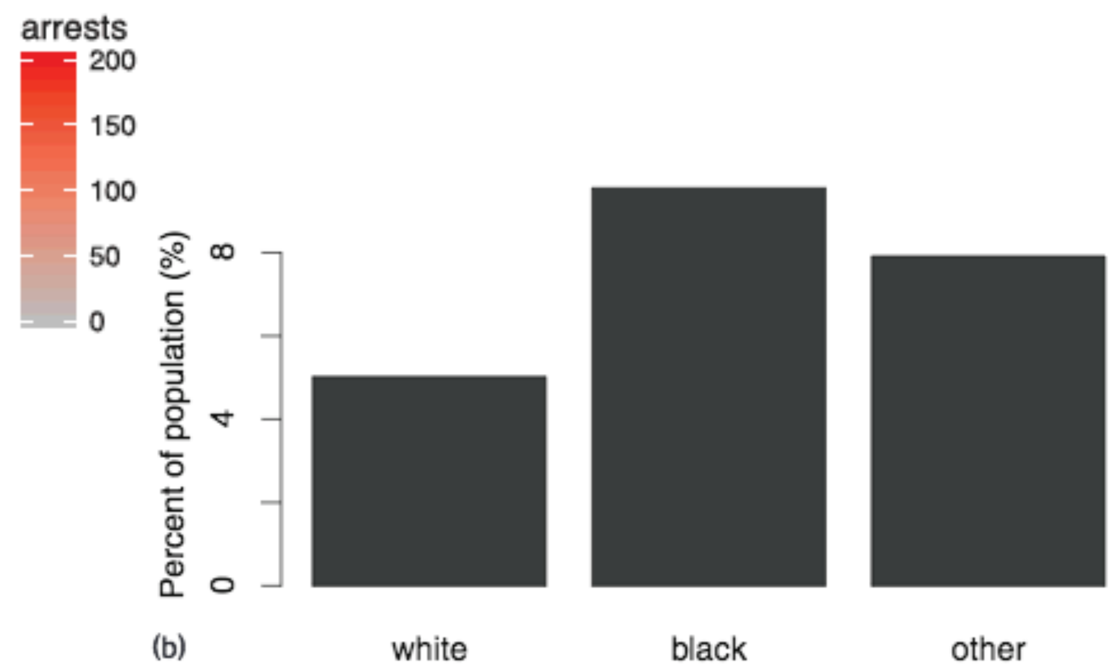
Targeted policing for drug crimes by race

Is my data biased? (histograms + geo)

[Lum, Isaac: *Significance, 2016*]



Number of drug arrests made by the Oakland, CA, police department in 2010



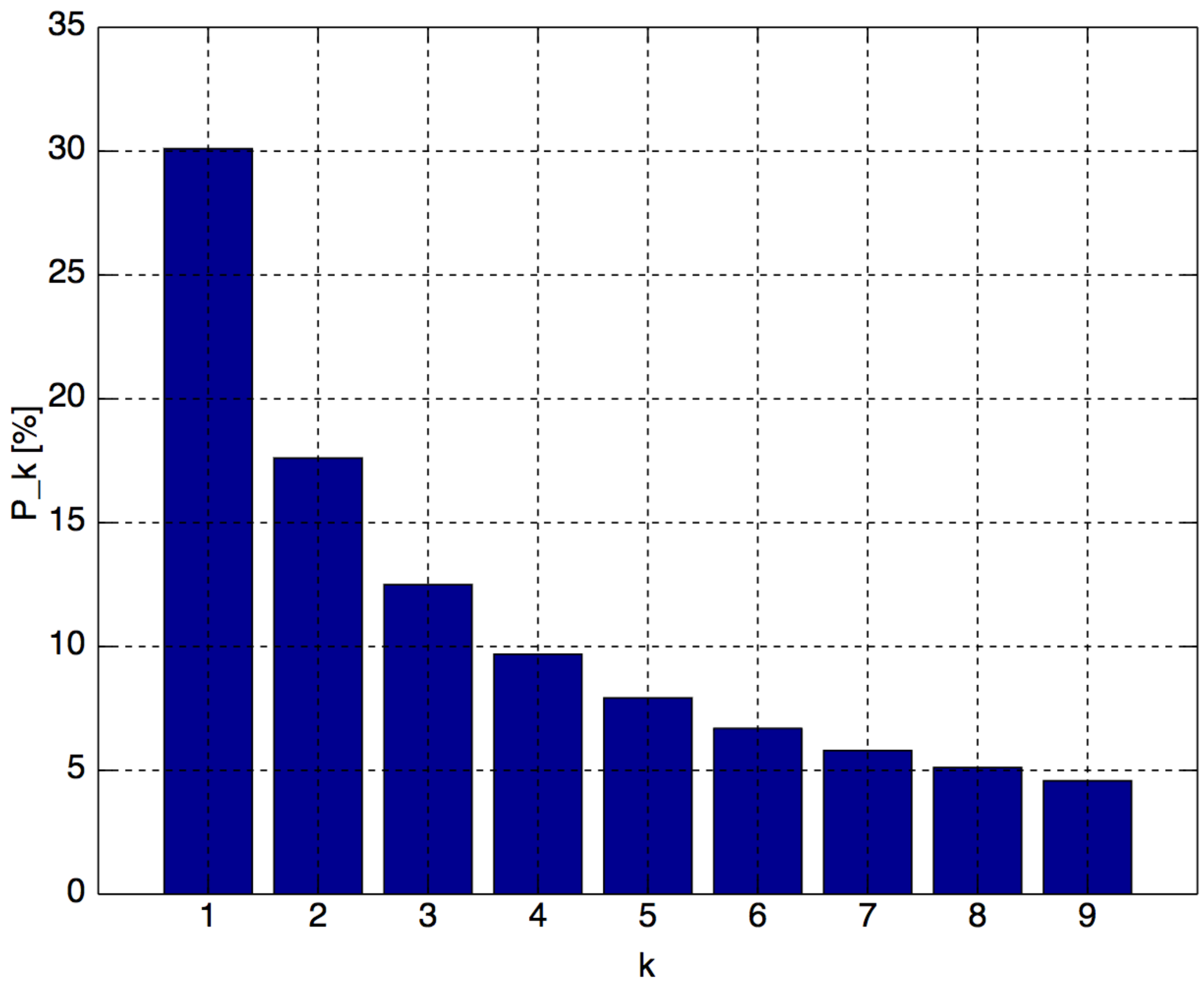
Targeted policing for drug crimes by race

Benford Law (first digit law)

[Benford: "The law of anomalous numbers" *Proc. Am. Philos. Soc., 1938*]

The distribution of the first digit **d** of a number, in many naturally occurring domains, approximately follows

$$P(d) = \log_{10}\left(1 + \frac{1}{d}\right)$$



1 is the most frequent leading digit, followed by 2, etc.

https://en.wikipedia.org/wiki/Benford%27s_law

Benford Law (first digit law)

[Benford: "The law of anomalous numbers" *Proc. Am. Philos. Soc., 1938*]

The distribution of the first digit **d** of a number, in many naturally occurring domains, approximately follows

$$P(d) = \log_{10}\left(1 + \frac{1}{d}\right)$$

Holds if log(x) is uniformly distributed. **Most accurate** when values are distributed across multiple orders of magnitude, especially **if the process generating the numbers is described by a power law** (common in nature)

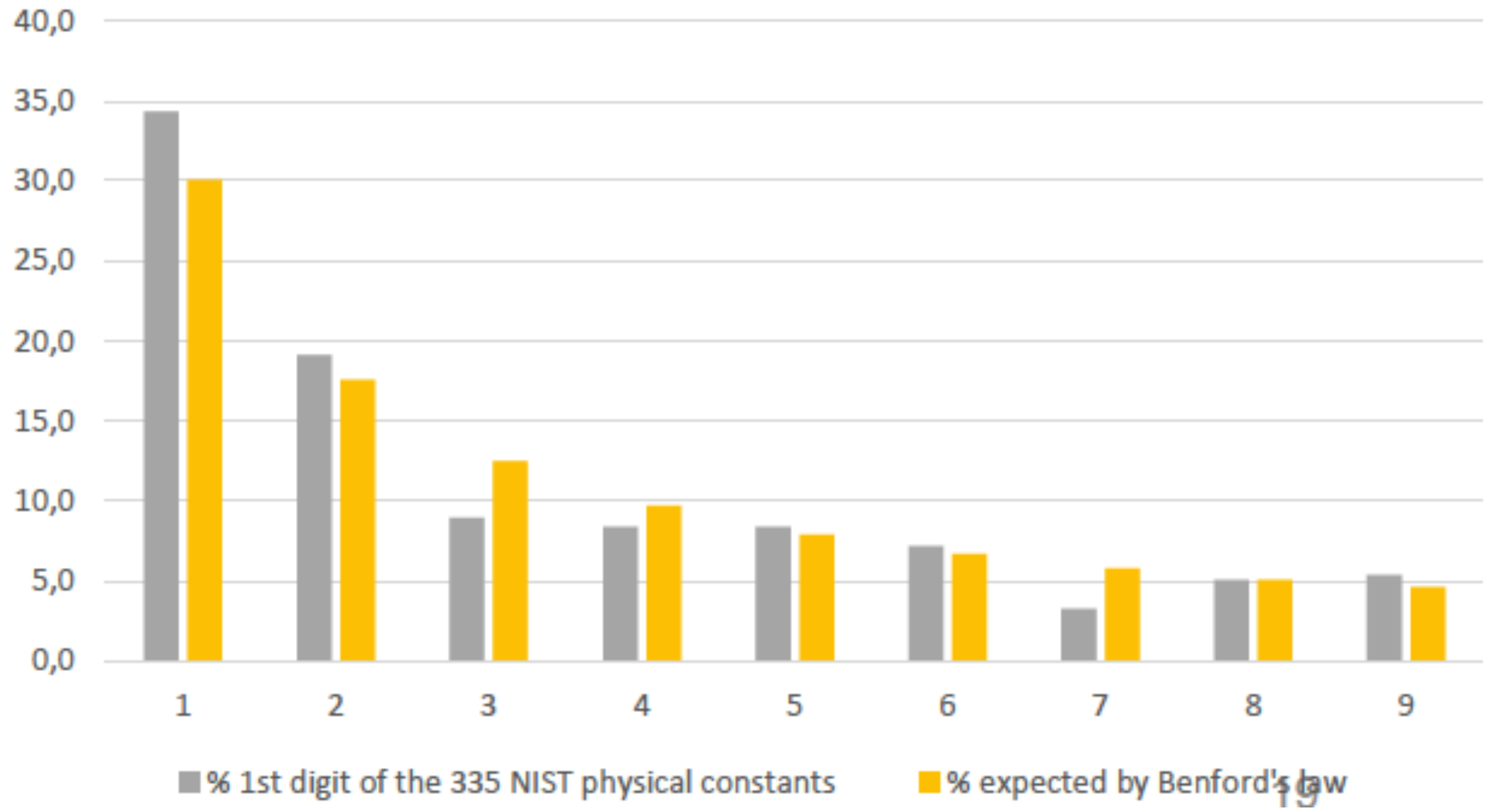
A logarithmic scale bar. Picking a random x position uniformly on this number line, roughly 30% of the time the first digit of the number will be 1.

https://en.wikipedia.org/wiki/Benford%27s_law

Benford Law: an example

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

height of tallest structures



Benford Law: other examples

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

- surface area of 355 rivers
- sizes of 3,259 US populations
- 104 physical constants
- 1,800 molecular weights
- 308 numbers contained in an issue of Reader's Digest
- Street addresses of the first 342 persons listed in American Men of Science
- ….

used in fraud detection!

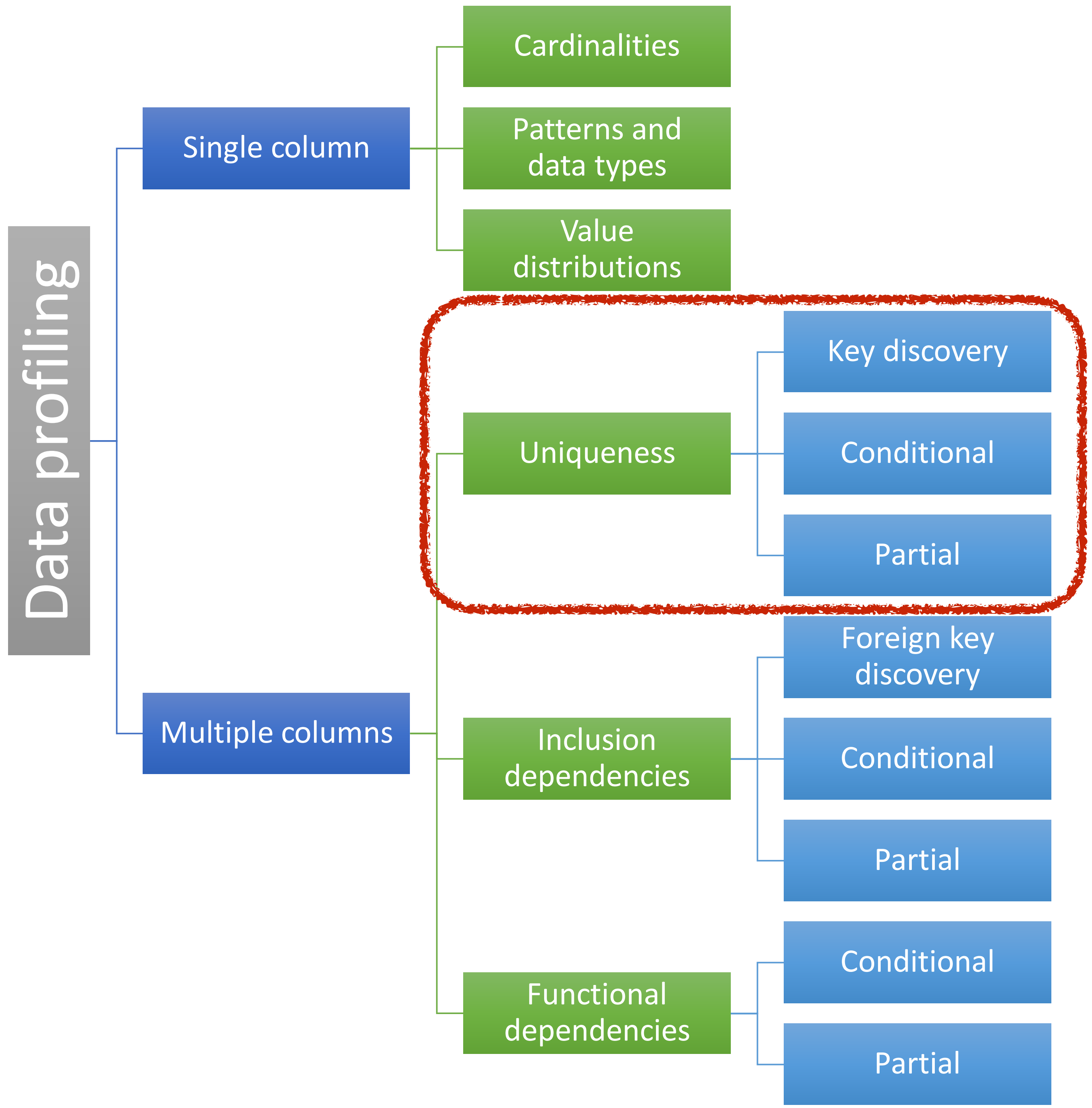
[Benford: "The law of anomalous numbers" *Proc. Am. Philos. Soc.*, *1938*]

Classification of data profiling tasks

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

UID	sex	race	MarriageSta	DateOfBirth	age	juv_fel_cour	decile_score
1	0	1	1	4/18/47	69	0	1
2	0	2	1	1/22/82	34	0	3
3	0	2	1	5/14/91	24	0	4
4	0	2	1	1/21/93	23	0	8
5	0	1	2	1/22/73	43	0	1
6	0	1	3	8/22/71	44	0	1
7	0	3	1	7/23/74	41	0	6
8	0	1	2	2/25/73	43	0	4
9	0	3	1	6/10/94	21	0	3
10	0	3	1	6/1/88	27	0	4
11	1	3	2	8/22/78	37	0	1
12	0	2	1	12/2/74	41	0	4
13	1	3	1	6/14/68	47	0	1
14	0	2	1	3/25/85	31	0	3
15	0	4	4	1/25/79	37	0	1
16	0	2	1	6/22/90	25	0	10
17	0	3	1	12/24/84	31	0	5
18	0	3	1	1/8/85	31	0	3
19	0	2	3	6/28/51	64	0	6
20	0	2	1	11/29/94	21	0	9
21	0	3	1	8/6/88	27	0	2
22	1	3	1	3/22/95	21	0	4
23	0	4	1	1/23/92	24	0	4
24	0	3	3	1/10/73	43	0	1
25	0	1	1	8/24/83	32	0	3
26	0	2	1	2/8/89	27	0	3
27	1	3	1	9/3/79	36	0	3

relational data (here: just one table)



An alternative classification

- To help understand the **statistics**, we look at value ranges, data types, value distributions per column or across columns, etc
- To help understand the **structure** - the (business) rules that generated the data - we look at unique columns / column combinations, dependencies between columns, etc - **reverse-engineer the relational schema** of the data we have
- We need both statistics and structure, they are mutually-reinforcing, and help us understand the **semantics** of the data - it's meaning

next up: relational model basics

The relational model

- Introduced by Edgar F. Codd in 1970 (Turing award)
- At the heart of relational database management systems (RDBMS)
  - a database consists of a collection of **relations** (tables)
  - **tuples** are stored in table rows
  - **attributes** of tuples are stored in table columns

	A	B	C	D	E	F	G	H
1	UID	sex	race	MarriageSta	DateOfBirth	age	juv_fel_cour	decile_score
2	1	0	1	1	4/18/47	69	0	1
3	2	0	2	1	1/22/82	34	0	3
4	3	0	2	1	5/14/91	24	0	4
5	4	0	2	1	1/21/93	23	0	8
6	5	0	1	2	1/22/73	43	0	1
7	6	0	1	3	8/22/71	44	0	1
8	7	0	3	1	7/23/74	41	0	6
9	8	0	1	2	2/25/73	43	0	4
10	9	0	3	1	6/10/94	21	0	3
11	10	0	3	1	6/1/88	27	0	4
12	11	1	3	2	8/22/78	37	0	1
13	12	0	2	1	12/2/74	41	0	4
14	13	1	3	1	6/14/68	47	0	1
15	14	0	2	1	3/25/85	31	0	3
16	15	0	4	4	1/25/79	37	0	1
17	16	0	2	1	6/22/90	25	0	10
18	17	0	3	1	12/24/84	31	0	5
19	18	0	3	1	1/8/85	31	0	3
20	19	0	2	3	6/28/51	64	0	6
21	20	0	2	1	11/29/94	21	0	9
22	21	0	3	1	8/6/88	27	0	2
23	22	1	3	1	3/22/95	21	0	4
24	23	0	4	1	1/23/92	24	0	4
25	24	0	3	3	1/10/73	43	0	1
26	25	0	1	1	8/24/83	32	0	3
27	26	0	2	1	2/8/89	27	0	3
28	27	1	3	1	9/3/79	36	0	3

The relational model

- Relations are **unordered collections** of tuples
  - conceptually, a relation is a **set** of tuples
  - however, SQL implements a relation as a **multiset** (bag) of tuples
- Why this model?
  - Simple yet powerful. Great for processing very large data sets in bulk

	A	B	C	D	E	F	G	H
1	UID	sex	race	MarriageSta	DateOfBirth	age	juv_fel_cour	decile_score
2	1	0	1	1	4/18/47	69	0	1
3	2	0	2	1	1/22/82	34	0	3
4	3	0	2	1	5/14/91	24	0	4
5	4	0	2	1	1/21/93	23	0	8
6	5	0	1	2	1/22/73	43	0	1
7	6	0	1	3	8/22/71	44	0	1
8	7	0	3	1	7/23/74	41	0	6
9	8	0	1	2	2/25/73	43	0	4
10	9	0	3	1	6/10/94	21	0	3
11	10	0	3	1	6/1/88	27	0	4
12	11	1	3	2	8/22/78	37	0	1
13	12	0	2	1	12/2/74	41	0	4
14	13	1	3	1	6/14/68	47	0	1
15	14	0	2	1	3/25/85	31	0	3
16	15	0	4	4	1/25/79	37	0	1
17	16	0	2	1	6/22/90	25	0	10
18	17	0	3	1	12/24/84	31	0	5
19	18	0	3	1	1/8/85	31	0	3
20	19	0	2	3	6/28/51	64	0	6
21	20	0	2	1	11/29/94	21	0	9
22	21	0	3	1	8/6/88	27	0	2
23	22	1	3	1	3/22/95	21	0	4
24	23	0	4	1	1/23/92	24	0	4
25	24	0	3	3	1/10/73	43	0	1
26	25	0	1	1	8/24/83	32	0	3
27	26	0	2	1	2/8/89	27	0	3
28	27	1	3	1	9/3/79	36	0	3

The relational model

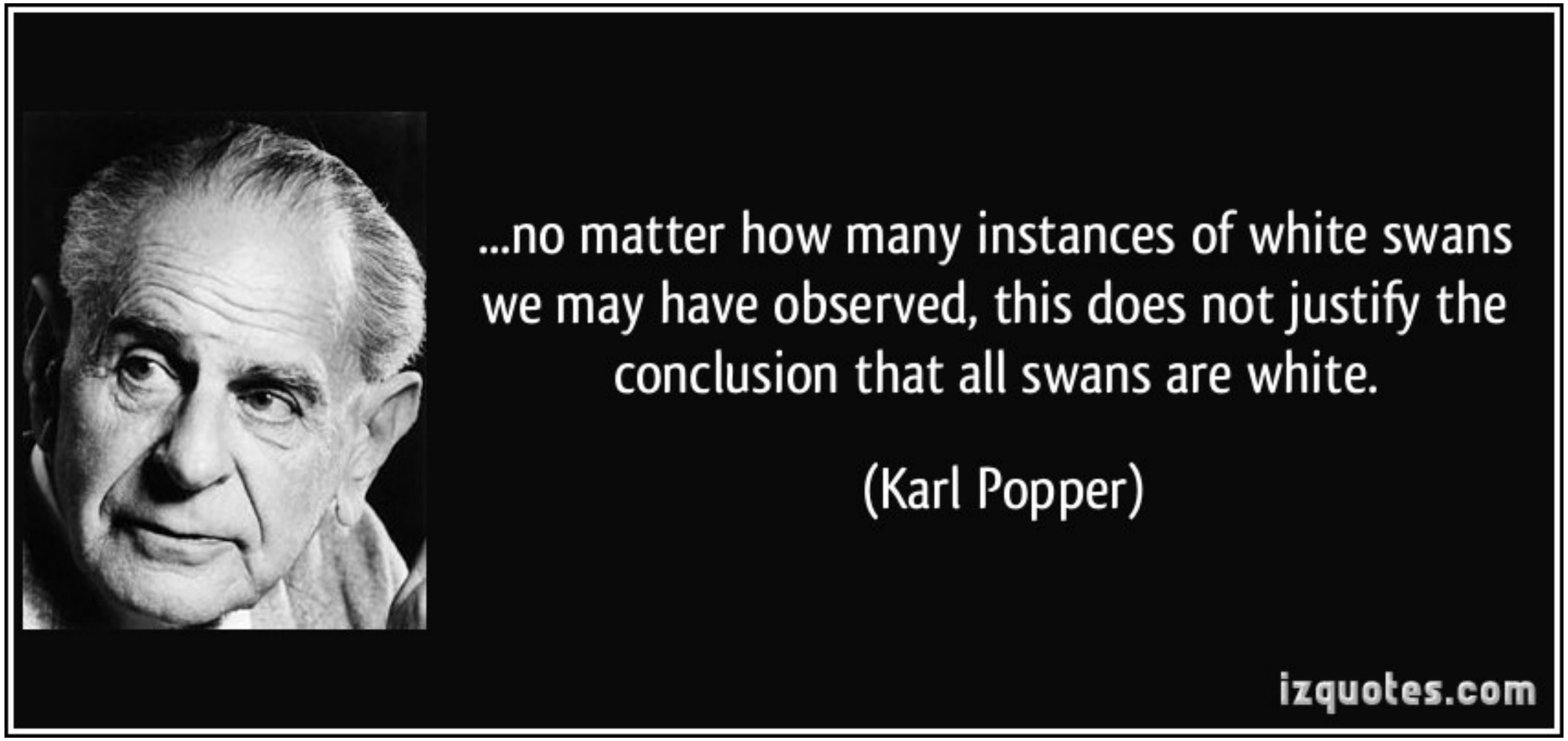
Episodes (<u>season</u>: int, <u>num</u>: int, title: string, viewers: long)

<u>season</u>	<u>num</u>	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M
2	2	The Night Lands	3.8 M

- **Relation**: a set or tuples - order doesn't matter, all tuples are distinct
- **Attribute**: a column in a relation (e.g., season)
- **Domain**: data type of an attribute (e.g., season: int)
- **Tuple**: a row in a relation, e.g., (1, 2, The Kingsroad, 2.2 M)

Schema vs. instances

Relation schema is a description of a relation in terms of relation name, attribute names, attribute datatypes, constraints (e.g., keys). A schema describes **all valid instances** of a relation.



Schema vs. instances

Relation schema is a description of a relation in terms of relation name, attribute names, attribute datatypes, constraints (e.g., keys). A schema describes **all valid instances** of a relation.

schema Episodes (<u>season</u>: integer, <u>num</u>: integer, title: string, viewers: integer)

instance 1

<u>season</u>	<u>num</u>	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M

instance 2

<u>season</u>	<u>num</u>	title	viewers
1	20	Blah, Blah and Blah	0
4	7	Yet Another Title	10 B

instance 3

<u>season</u>	<u>num</u>	title	viewers

Integrity constraints

- Ensure that data adheres to the rules of the application
- Specified **when schema is defined**
- Checked and enforced by the DBMS when relations are modified (tuples added / removed / updated)
- Must **hold on every valid instance** of the database

1. **Domain constraints** - specify valid data types for each attribute, e.g., Students (sid: integer, name: string, gpa: decimal)
2. **Key constraints** - define a unique identifier for each tuple
3. **Referential integrity constraints** - specify links between tuples
4. **Functional dependencies** - show relationships within a table

Key constraints

A set of attributes is a **candidate key** for a relation if:
(1) no two distinct tuples can have the same values for all key attributes (candidate key **uniquely identifies** a tuple), *and*
(2) this is not true for any subset of the key attributes (candidate key **is minimal**)

- If condition (2) is not met, we have a **superkey**
- There may be more than one candidate key for a relation, if so, one is designated as the **primary key**
- All candidate key should be known to property enforce data integrity

Example: name possible candidate keys

Students (sid: integer, login: string, name: string, dob: date)

Key constraints

Example: Students (sid: integer, login: string, name: string, dob: date)

three possible SQL implementations

```
create table Students (
   sid   integer      primary key,
   login varchar(128) unique,
   name  varchar(128),
   dob   date,
   gpa   decimal,
   unique (name, dob) );
```

```
create table Students (
   sid   integer      unique,
   login varchar(128) primary key,
   name  varchar(128),
   dob   date,
   gpa   decimal,
   unique (name, dob) );
```

```
create table Students (
   sid   integer       unique,
   login varchar(128)  unique,
   name  varchar(128),
   dob   date,
   gpa   decimal,
   primary key (name, dob) );
```

NB: every relation must have exactly one primary key

DB 101: Where do business rules come from?

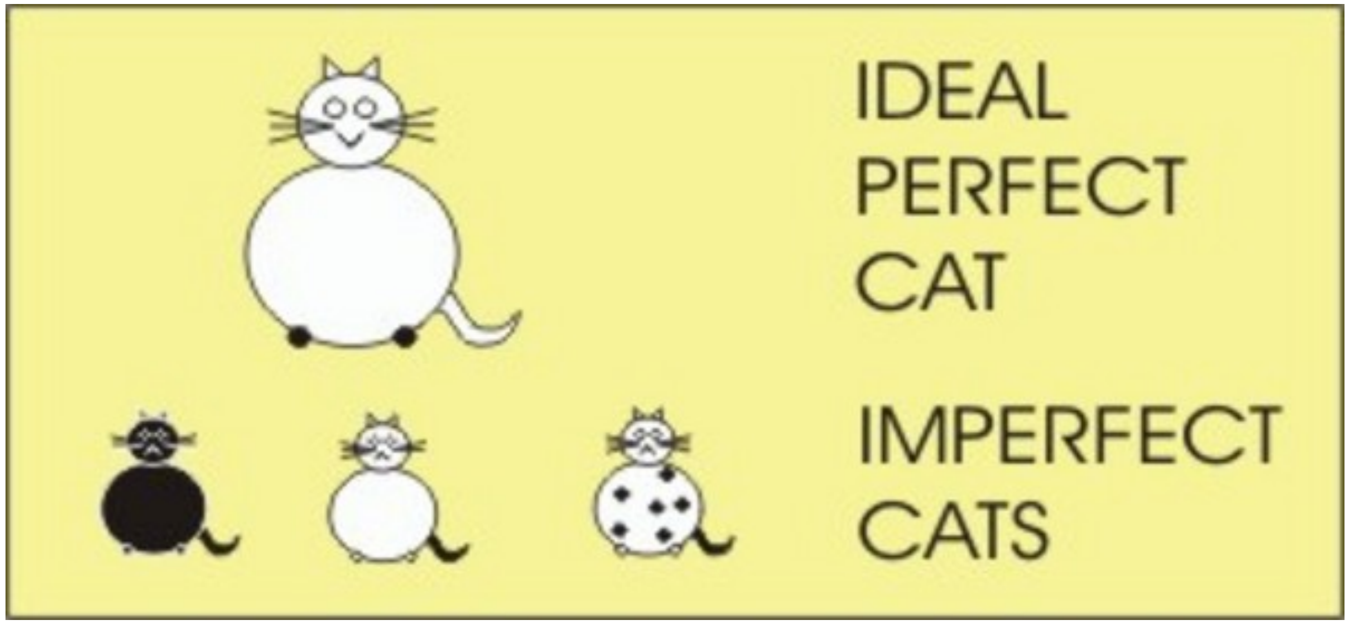
- **Business rules are given**: by the client, by the application designer, by your boss
- Once you know the rules, you create a **relational schema** that encodes these business rules (sometimes starting with the entity-relationship model, sometimes with the relational model directly)
- We can **never-ever-ever deduce business rules by looking at an instance** of a relation!
- We can sometimes know which rules do not hold, but we cannot be sure which rules do hold

Employee

id	login	name
1	jim	Jim Morrison
2	amy	Amy Winehouse
3	amy	Amy Pohler
4	raj	Raj Kapoor

1. Which column **is not** a candidate key?
2. Which column(s) **may be** a candidate key?
3. Give 2 create table statements for which this instance is valid.

DB (databases) vs. DS (data science)



https://midnightmediamusings.wordpress.com/2014/07/01/plato-and-the-theory-of-forms/

- **DB**: start with the schema, admit only data that fits; iterative refinement is possible, and common, but we are still schema-first
- **DS**: start with the data, figure out what schema it fits, or almost fits - reasons of usability, repurposing, low start-up cost

the “right” approach is somewhere between these two, data profiling aims to bridge between the two worlds / points of view / methodologies

Discovering uniques

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** (or a **"unique"** for short) is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

$Episodes(season, num, title, viewers)$

season	num	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M

Projection is a relational algebra operation that takes as input relation ***R*** and returns a new relation ***R'*** with a subset of the columns of ***R***.

$\pi_{season}(Episodes)$

season
1
1
2

$\pi_{season,num}(Episodes)$

season	num
1	1
1	2
2	1

$\pi_{title}(Episodes)$

title
Winter is Coming
The Kingsroad
The North Remembers

Discovering uniques

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** (or a **"unique"** for short) is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

Episodes(season,num,title,viewers)

season	num	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M

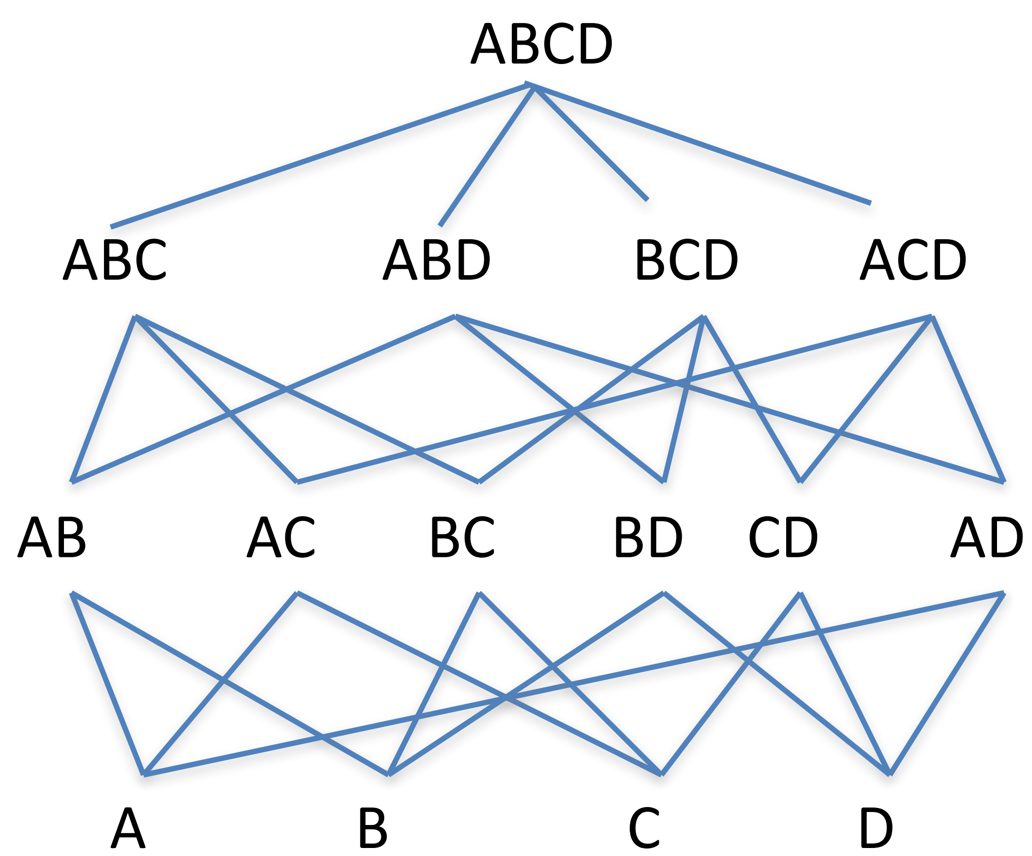
Projection is a relational algebra operation that takes as input relation ***R*** and returns a new relation ***R'*** with a subset of the columns of ***R***.

- Recall that more than one set of attributes **X** may be unique
- It may be the case that **X** and **Y** are both unique, and that they are not disjoint. When is this interesting?

Discovering uniques

R (A, B, C, D)

attribute lattice of ***R***



$$\binom{4}{1} = 1$$

$$\binom{4}{3} = 4$$

$$\binom{4}{2} = 6$$

$$\binom{4}{4} = 4$$

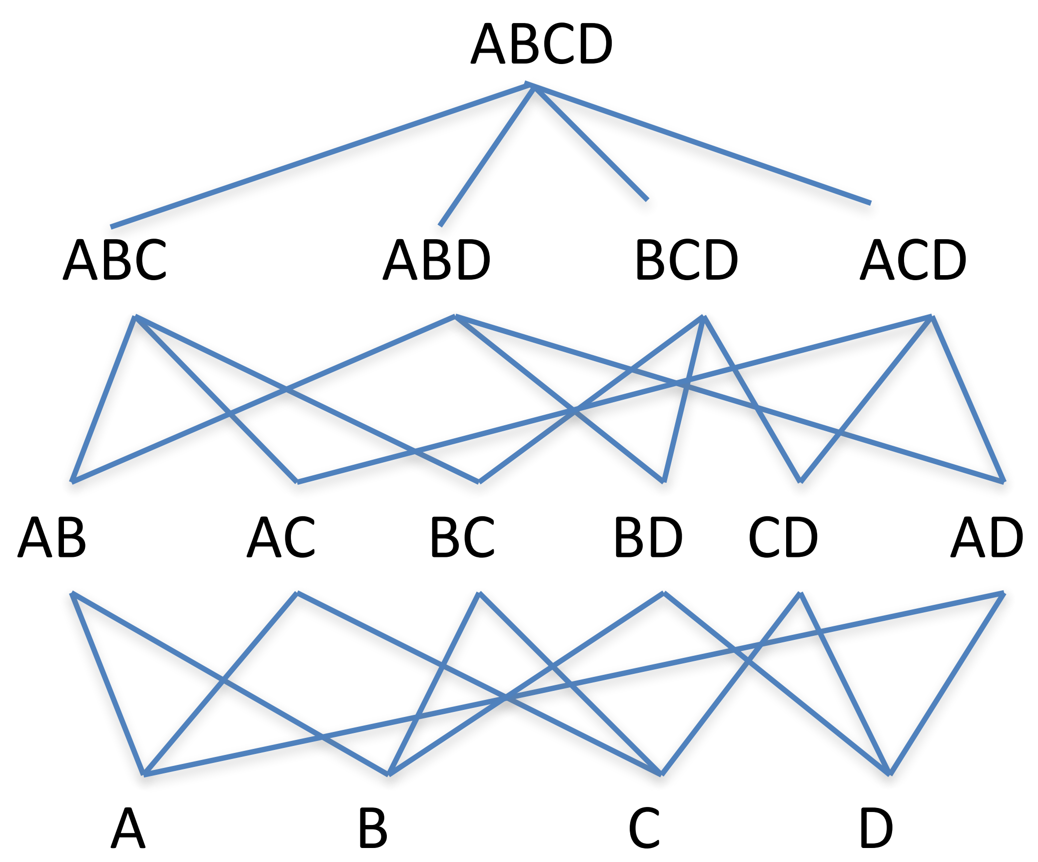
What's the size of the attribute lattice of ***R***?

Look at all attribute combinations?

Discovering uniques

R (A, B, C, D)

attribute lattice of R



- If **X** is unique, then what can we say about its **superset Y**?
- If **X** is non-unique, then what can we say about its **subset Z**?

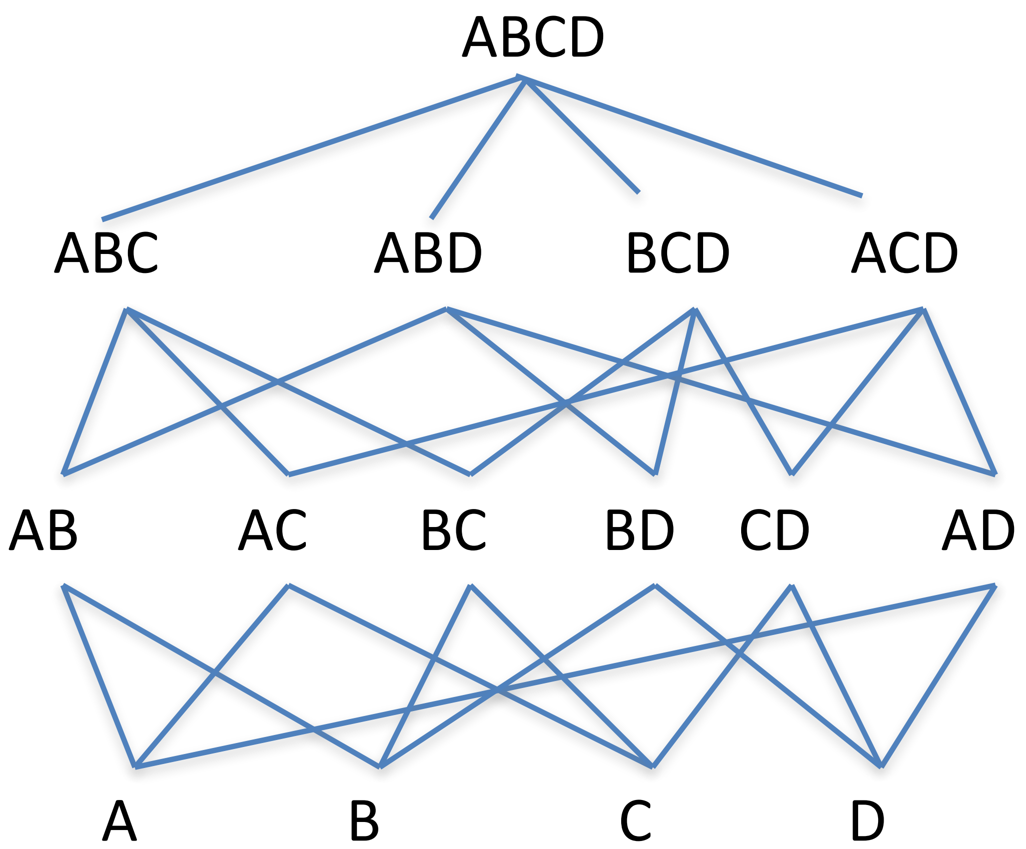
Discovering uniques

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** (or a **“unique”** for short) is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a set of attributes ***Y*** is **non-unique** if its projection contains duplicates in ***r***

X is **minimal unique** if every subset ***Y*** of ***X*** is non-unique

Y is maximal non-unique if every superset ***X*** of ***Y*** is unique



From uniques to candidate keys

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

$$Episodes(season, num, title, viewers)$$

season	num	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M

A set of attributes is a **candidate key** for a relation if:

(1) no two distinct tuples can have the same values for all key attributes (candidate key **uniquely identifies** a tuple), *and*

(2) this is not true for any subset of the key attributes (candidate key **is minimal**)

A minimal unique of a relation instance is a (possible) candidate key of the relation schema. To find all possible candidate keys, find all minimal uniques in a relation instance.

Pivot: Frequent itemsets & association rules

- Problem formulation due to Agrawal, Imielinski, Swami, SIGMOD 1993
- Solution: the **Apriori** algorithm by Agrawal & Srikant, VLDB 1994
- Initially for **market-basket data** analysis, has many other applications, we'll see one today
- We wish to answer two related questions:
  - **Frequent itemsets:** Which items are often purchased together, e.g., milk and cookies are often bought together
  - **Association rules:** Which items will likely be purchased, based on other purchased items, e.g., if diapers are bought in a transaction, beer is also likely bought in the same transaction

Market-basket data

- $\boldsymbol{I = \{i_1, i_2, \ldots, i_m\}}$ is the set of available items, e.g., a product catalog of a store
- $\boldsymbol{X} \subseteq \boldsymbol{I}$ is an **itemset**, e.g., {milk, bread, cereal}
- **Transaction** $\boldsymbol{t}$ is a set of items purchased together, $\boldsymbol{t} \subseteq \boldsymbol{I}$, has a transaction id (TID)

  $\boldsymbol{t_1}$: {bread, cheese, milk}

  $\boldsymbol{t_2}$: {apple, eggs, salt, yogurt}

  $\boldsymbol{t_3}$: {biscuit, cheese, eggs, milk}

- Database $\boldsymbol{T}$ is a set of transactions $\boldsymbol{\{t_1, t_2, \ldots, t_n\}}$
- A transaction $\boldsymbol{t}$ **supports** an itemset $\boldsymbol{X}$ if $\boldsymbol{X} \subseteq \boldsymbol{t}$
- Itemsets supported by at least ***minSupp*** transactions are called **frequent itemsets**

minSupp, which can be a number or a percentage, is specified by the user

Itemsets

TID	Items
1	A
2	A C
3	A B D
4	A C
5	A B C
6	A B C

minSupp = 2 transactions

How many possible itemsets are there (excluding the empty itemset)?

$2^4 - 1 = 15$

	itemset	support
★	A	6
★	B	3
★	C	4
	D	1
★	A B	3
★	A C	4
	A D	1
★	B C	2
	B D	1
	C D	0
★	A B C	2
	A B D	1
	B C D	0
	A C D	0
	A B C D	0

Association rules

An association rule is an implication $X \rightarrow Y$, where $X,\ Y \subset I$, and $X \cap Y = \emptyset$

example: {milk, bread}$\rightarrow${cereal}

"A customer who purchased X is also likely to have purchased Y in the same transaction"

we are interested in rules with a **single item** in Y

can we represent {milk, bread} $\rightarrow$ {cereal, cheese}?

Rule $X \rightarrow Y$ holds with **support** *supp* in T if *supp* of transactions contain $X \cup Y$

Rule $X \rightarrow Y$ holds with confidence *conf* in T if *conf* % of transactions that contain X also contain Y

$conf \approx \Pr(Y \mid X)$

$conf\ (X \rightarrow Y) = supp\ (X \cup Y)\ /\ supp\ (X)$

Association rules

minSupp = 2 transactions
minConf = 0.75

	supp = 3
$A \rightarrow B$	conf = 3 / 6 = 0.5
$B \rightarrow A$	conf = 3 / 3 = 1.0 ★

	supp = 2
$B \rightarrow C$	conf = 2 / 3 = 0.67
$C \rightarrow B$	conf = 2 / 4 = 0.5

	supp = 4
$A \rightarrow C$	conf = 4 / 6 = 0.67
$C \rightarrow A$	conf = 4 / 4 = 1.0 ★

	supp = 2
$AB \rightarrow C$	conf = 2 / 3 = 0.67
$AC \rightarrow B$	conf = 2 / 4 = 0.5
$BC \rightarrow A$	conf = 2 / 2 = 1.0 ★

	itemset	support
★	A	6
★	B	3
★	C	4
	D	1
★	A B	3
★	A C	4
	A D	1
★	B C	2
	B D	1
	C D	0
★	A B C	2
	A B D	1
	B C D	0
	A C D	0
	A B C D	0

conf $(X \rightarrow Y)$ = ***supp*** (X U Y) / ***supp*** (X)

Association rule mining

- Goal: find all association rules that satisfy the user-specified minimum support and minimum confidence

- Algorithm outline
  - Step 1: find all frequent itemsets
  - Step 2: find association rules

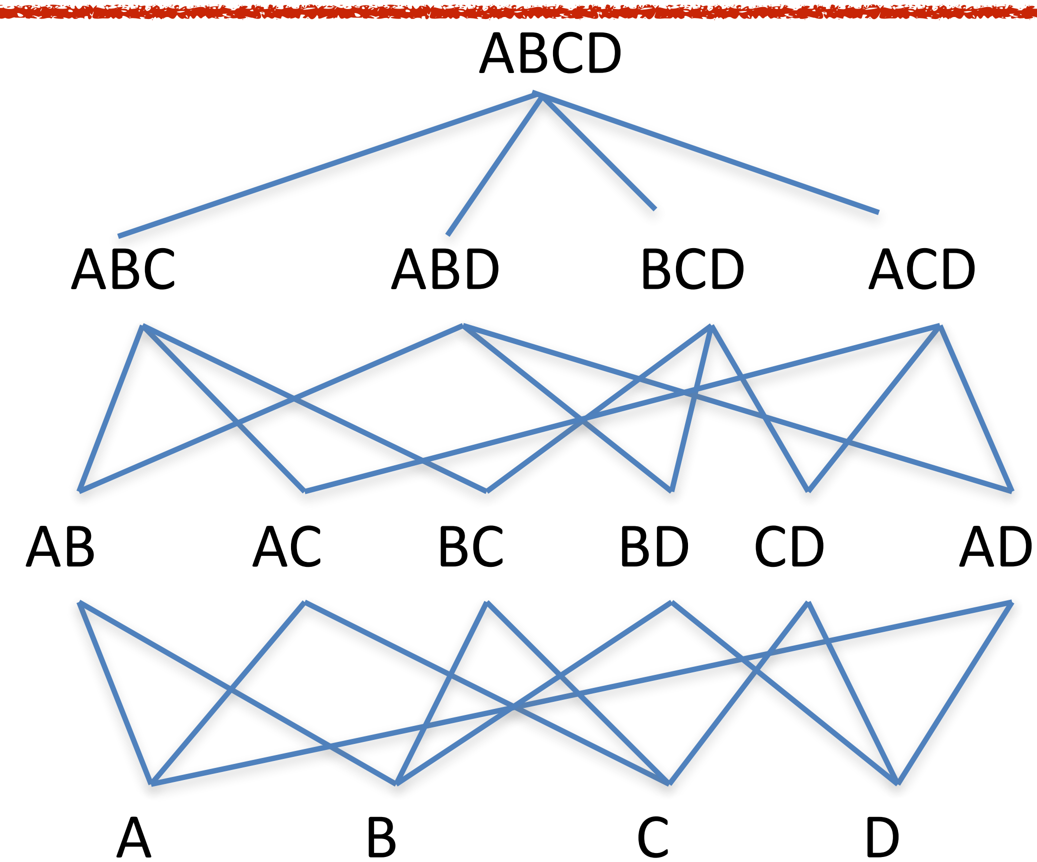
- Take 1: naïve algorithm for frequent itemset mining
  - Enumerate all subsets of ***I***, check their support in ***T***
  - **What is the complexity?**

Key idea: downward closure

itemset	support
★ A	6
★ B	3
★ C	4
D	1
★ A B	3
★ A C	4
A D	1
★ B C	2
B D	1
C D	0
★ A B C	2
A B D	1
B C D	0
A C D	0
A B C D	0

All subsets of a frequent itemset ***X*** are themselves frequent

So, if some subset of X is infrequent, then X cannot be frequent, we know this **apriori**



The converse is not true! If all subsets of ***X*** are frequent, ***X*** is not guaranteed to be frequent

The Apriori algorithm

Algorithm Apriori(*T*, *minSupp*)

F_1 = {*frequent 1-itemsets*};

for (k = 2; $F_{k-1} \neq \emptyset$; k++) **do**

$C_k \leftarrow$ **candidate-gen**(F_{k-1});

for each transaction $t \in T$ **do**

for each candidate $c \in C_k$ **do**

if c is contained in t **then**

c.count++;

end

end

$F_k \leftarrow \{c \in C_k \mid c.count \geq minSupp\}$

end

return $F \leftarrow \bigcup_k F_k$;

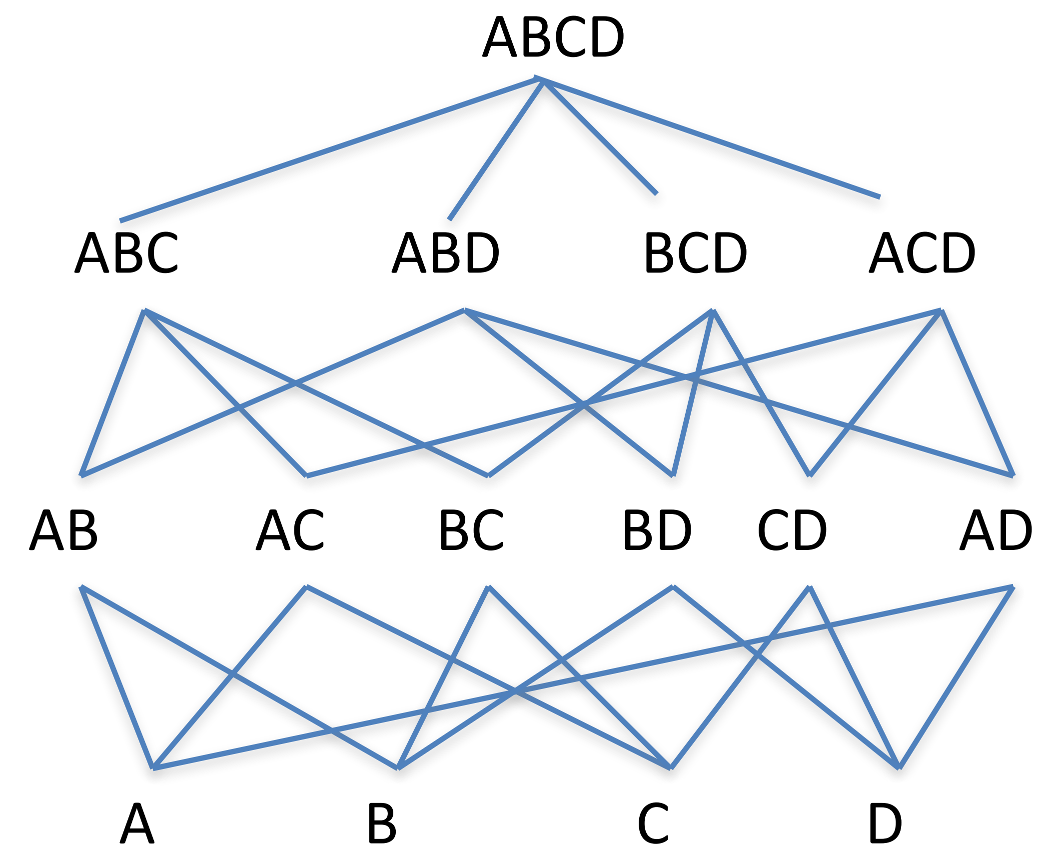
	itemset	support
★	A	6
★	B	3
★	C	4
	D	1
★	A B	3
★	A C	4
	A D	1
★	B C	2
	B D	1
	C D	0
★	A B C	2
	A B D	1
	B C D	0
	A C D	0
	A B C D	0

Candidate generation

The **candidate-gen** function takes F_{k-1} and returns a superset (called the candidates) of the set of all frequent k-itemsets. It has two steps:

Join: generate all possible candidate itemsets C_k of length k

Prune: optionally remove those candidates in C_k that have infrequent subsets



NYU

Candidate generation

Assume a lexicographic ordering of the items

Join

```
Insert into Ck (
  select   p.item1, p.item2, …, p.itemk-1, q.itemk-1
  from     Fk-1 p, Fk-1 q
  where    p.item1 = q.item1
  and      p.item2 = q.item2
  and      …
  and      p.itemk-1 < q.itemk-1 )
```

why not $p.item_{k-1} \neq q.item_{k-1}$?

Prune

for each c in C_k **do**

for each (k-1) subset s of c **do**

if (s not in F_{k-1}) **then**

delete c from C_k

Generating association rules

Rules $= \varnothing$

for each frequent *k-itemset* X **do**

    for each 1-itemset A $\subset$ X **do**

      compute conf (X / A → A) = supp(X) / sup (X / A)

      if conf (X / A → A) $\geq$ minConf **then**

        Rules $\leftarrow$ "X / A → A"

    end

  end

end

return Rules

Performance of *Apriori*

- The possible number of frequent itemsets is exponential, O($\boldsymbol{2^m}$), where $\boldsymbol{m}$ is the number of items
- Apriori exploits sparseness and locality of data
  - Still, it may produce a large number of rules: thousands, tens of thousands, ….
  - So, thresholds should be set carefully. What are some good heuristics?

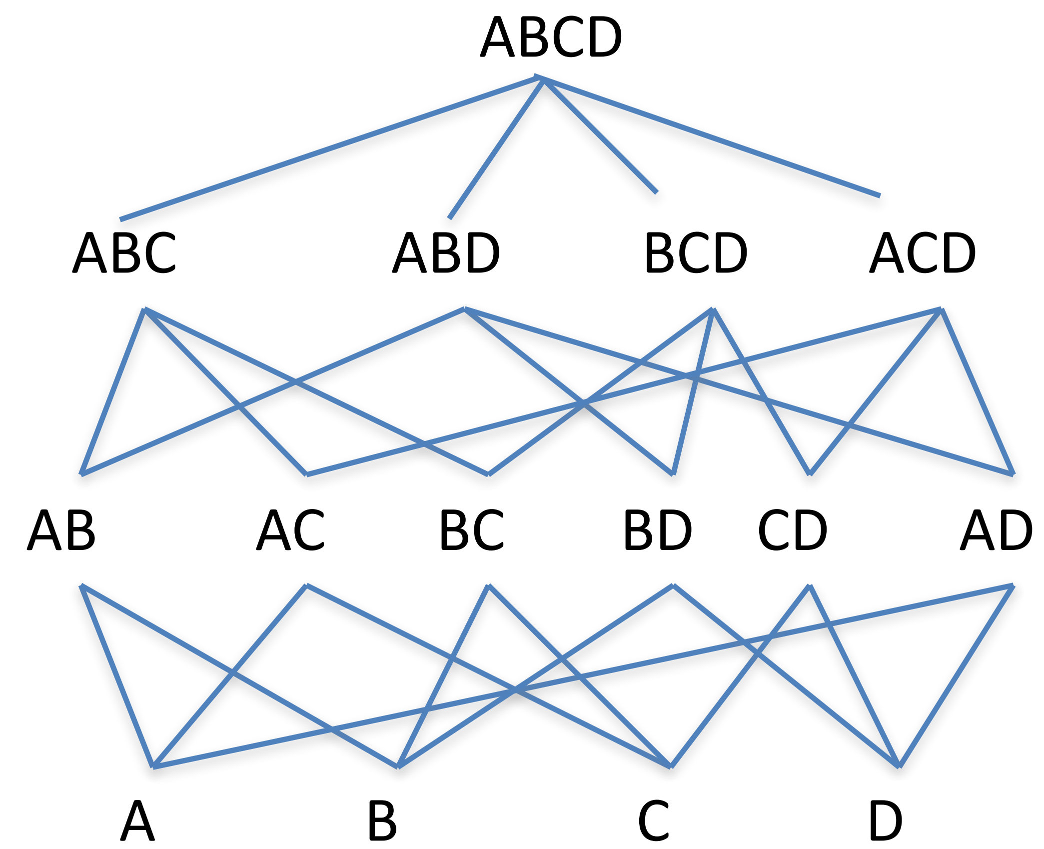
Back to data profiling: Discovering uniques

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** (or a **"unique"** for short) is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

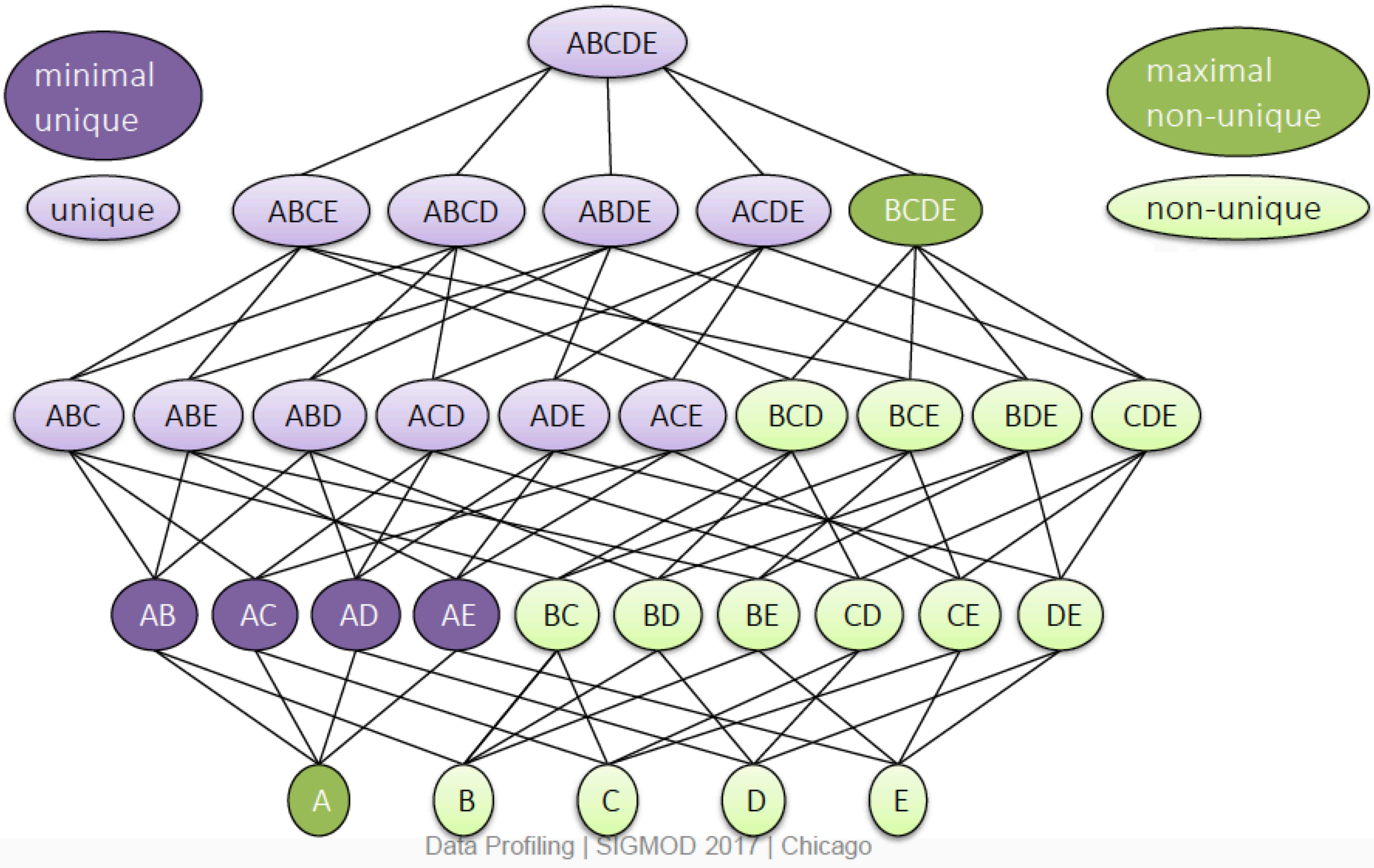
Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a set of attributes ***Y*** is **non-unique** if its projection contains duplicates in ***r***

X is **minimal unique** if every subset ***Y*** of ***X*** is non-unique

Y is maximal non-unique if every superset ***X*** of ***Y*** is unique



Output


minimal unique
unique
maximal non-unique
non-unique
ABCDE
ABCE
ABCD
ABDE
ACDE
BCDE
ABC
ABE
ABD
ACD
ADE
ACE
BCD
BCE
BDE
CDE
AB
AC
AD
AE
BC
BD
BE
CD
CE
DE
A
B
C
D
E
Data Profiling | SIGMOD 2017 | Chicago

From uniques to candidate keys

Given a relation schema ***R*** *(A, B, C, D)* and a relation instance ***r***, a **unique column combination** is a set of attributes ***X*** whose **projection** contains no duplicates in ***r***

$$Episodes(season, num, title, viewers)$$

season	num	title	viewers
1	1	Winter is Coming	2.2 M
1	2	The Kingsroad	2.2 M
2	1	The North Remembers	3.9 M

A set of attributes is a **candidate key** for a relation if:

(1) no two distinct tuples can have the value values for all key attributes (candidate key **uniquely identifies** a tuple), *and*

(2) this is not true for any subset of the key attributes (candidate key **is minimal**)

A minimal unique of a relation instance is a (possible) candidate key of the relation schema. To find such possible candidate keys, find all minimal uniques in a given relation instance.

Apriori-style uniques discovery

[Abedjan, Golab, Naumann; *SIGMOD 2017*]

A minimal unique of a relation instance is a **(possible) candidate key** of the relation schema.

Algorithm Uniques **// sketch, similar to HCA**

U_1 = {1-uniques} N_1 = {1-non-uniques}

for (k = 2; $N_{k-1} \neq \varnothing$; k++) **do**

$C_k \leftarrow$ **candidate-gen**(N_{k-1})

$U_k \leftarrow$ **prune-then-check** (C_k)

// prune candidates with unique sub-sets, and with **value distributions that cannot be unique**

// check each candidate in pruned set for uniqueness

$N_k \leftarrow C_k \setminus U_k$

end

return $U \leftarrow \bigcup_k U_k$;

breadth-first bottom-up strategy for attribute lattice traversal