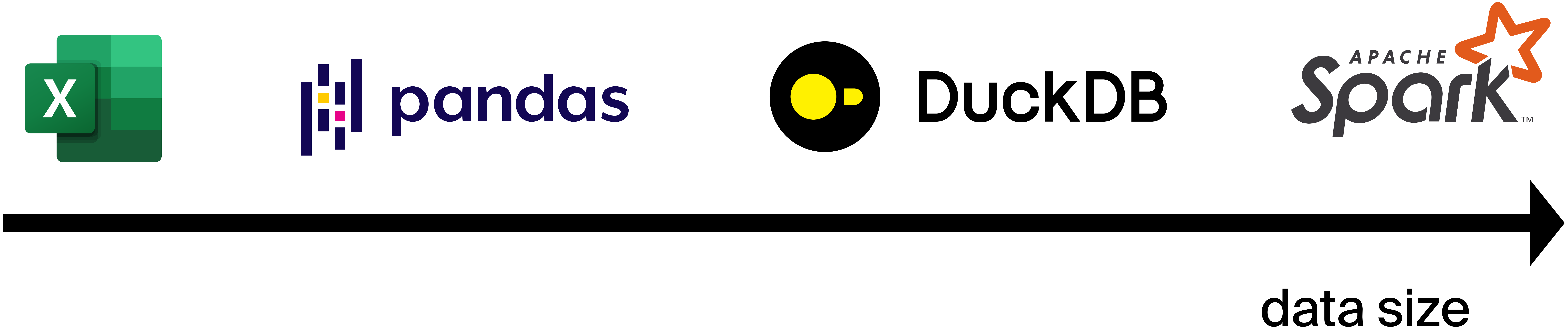
Advanced Data Management (CSCI 640/490)

Data Wrangling

Dr. David Koop

Northern Illinois University

Data Processing Systems



[G. Szárnyas, 2024]

DuckDB is In-Process, OLAP DBMS

	Transactional	Analytical
In-process	SQLite	DuckDB
Client–server	MySQL, PostgreSQL	snowflake, VERTICA

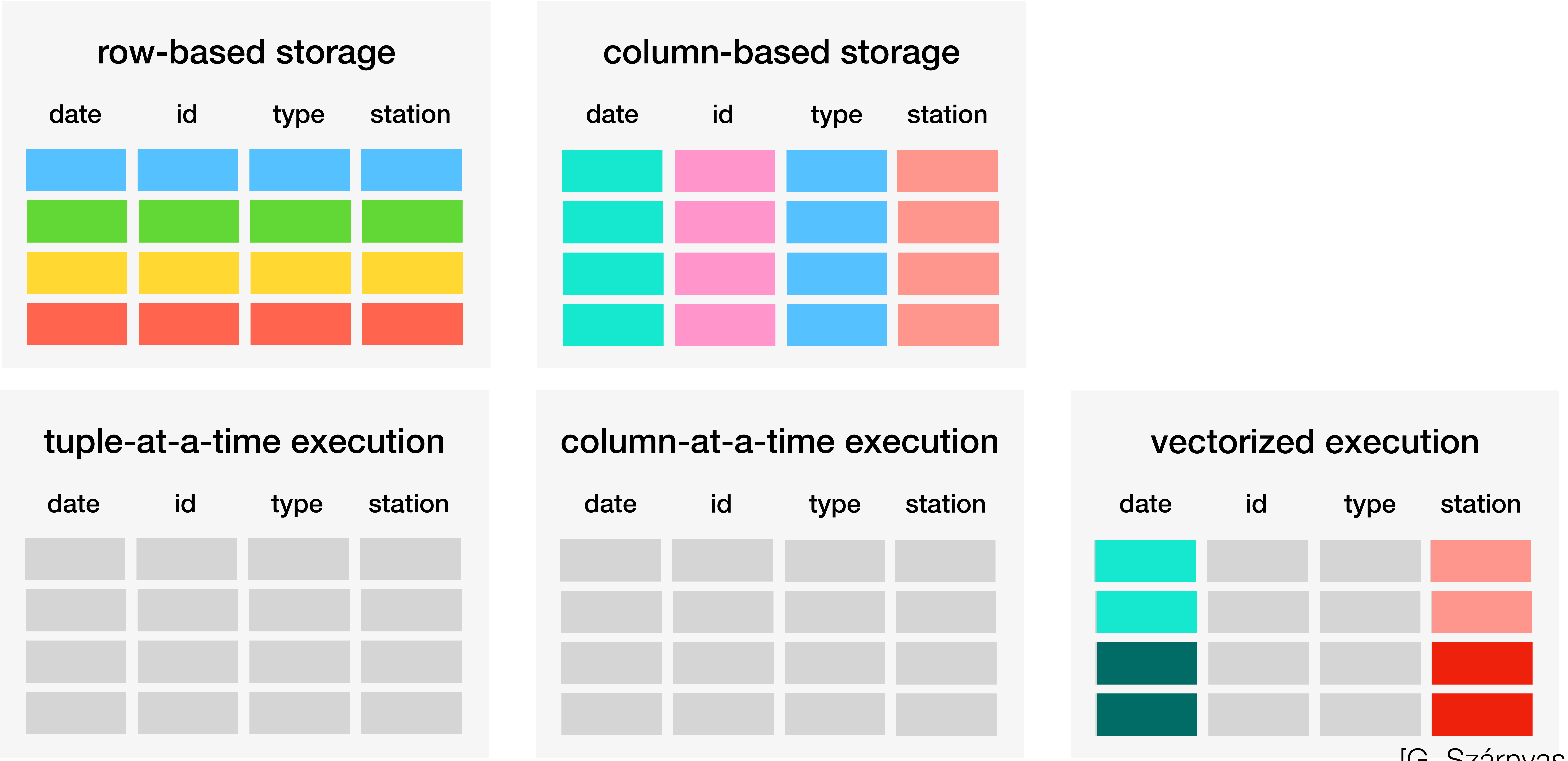
[G. Szárnyas, 2024]

DuckDB Characteristics

- Columnar Data Storage: Compare with row-based, compression
- Vectorized Execution Engine: Optimized for CPU Cache Locality
- End-to-end Query Optimization: Expression rewriting, Join Ordering, Subquery Flattening, Filter/Projection Pushdown
- Automatic Parallelism: Scanners, Aggregations, Joins
- Data Compression: Linked with Columnar Data
- Beyond Memory Execution: Supports data that does not fit into memory, graceful degradation

[P. Holanda, 2023]

Columnar Storage & Vectorized Execution



[G. Szárnyas, 2024]

Zonemaps

- Zonemaps are min/max indices
- Exist for **each column** in **each row group**

row group 1

date	id	type	station
May 30	1	Intercity	Ams C
May 30	2	Sprinter	Utrecht
May 31	3	ICE Intl	Ams C
May 31	4	Intercity	Schiphol

	date	id	type	station
min	May 30	1	ICE Intl	Ams C
max	May 31	4	Sprinter	Utrecht

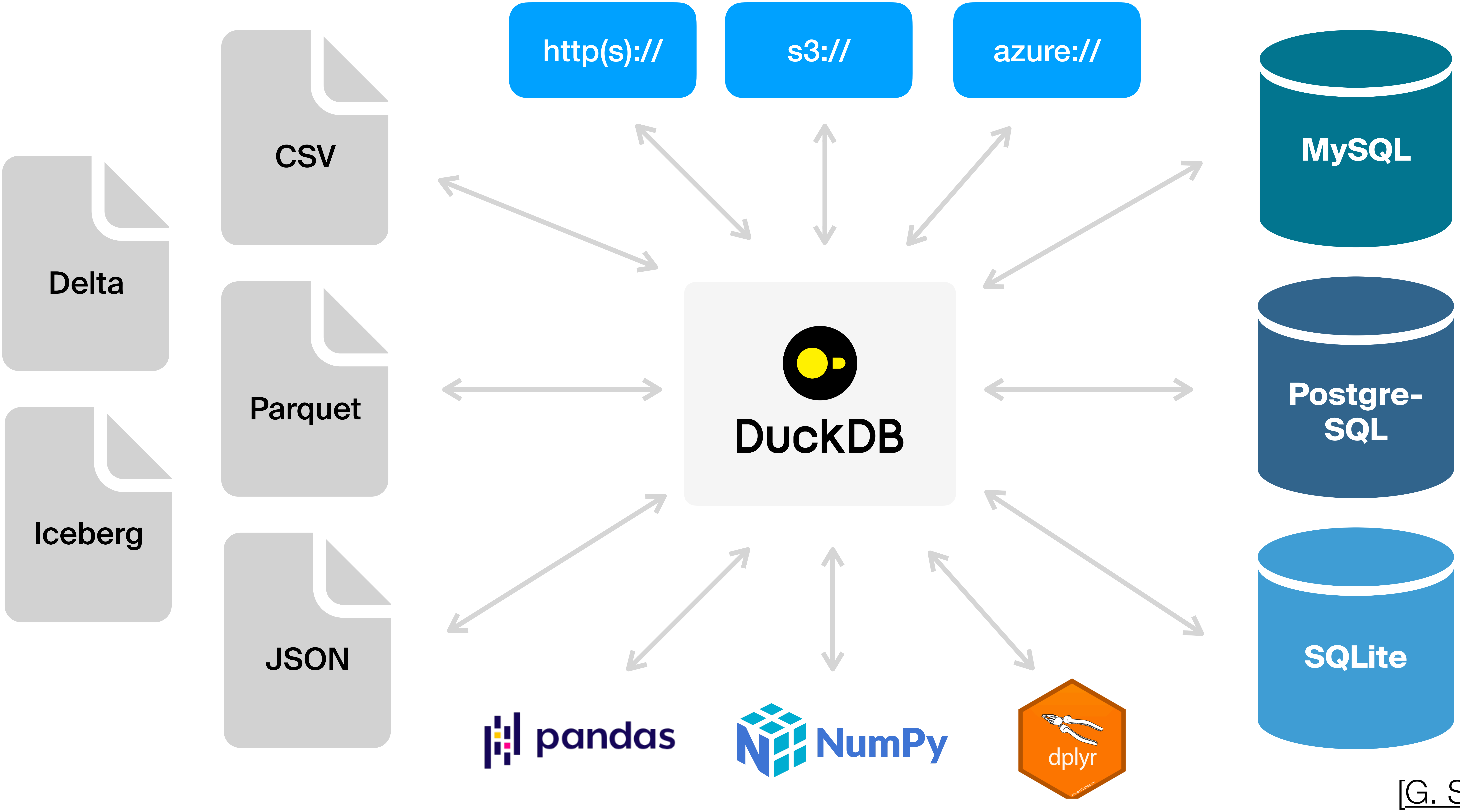
row group 2

date	id	type	station
June 1	5	Sprinter	Schiphol
June 1	6	Sprinter	Utrecht
June 1	7	Intercity	Utrecht
June 2	8	Intercity	Ams C

	date	id	type	station
min	June 1	5	Intercity	Ams C
max	June 2	8	Sprinter	Utrecht

[G. Szárnyas, 2024]

DuckDB Supported Formats and Protocols



[G. Szárnyas, 2024]

More DuckDB Characteristics

- Portable: Written in C++11, Inlined Dependencies, APIs for most languages
- Open Source: MIT License, Built on Open Standards
- Limitations:
  - Concurrency Control uses MVCC and WAL but not good for OLTP-heavy workloads
  - Execution: **single-node**, not distributed execution, scales to large nodes

Chicago Food Inspections Exploration

- Using Pandas
- Using DuckDB
- Using Polars

Courselets

- Deeper dive into the functionality for each of polars, DuckDB, and pandas
- Contain interactive examples as well as exercises
- Opportunity to provide feedback

Assignment 2

- Assignment 1 Questions with polars, DuckDB, and pandas
- CS 640 students do all, CS 490 do polars & DuckDB (pandas is EC)
- Can work by framework or by query
- Most questions can be answered with a single statement… but that statement can take a while to write
  - Read documentation
  - Check hints

Data

Data

- What is data?
  - Types
  - Semantics
- How is data structured?
  - Tables (Data Frames)
  - Databases
  - Data Cubes
- What formats is data stored in?
- Raw versus derived data

Data

- What is this data?

R011	42ND STREET & 8TH AVENUE	00228985	00008471	00000441	00001455	00000134	00033341	00071255
R170	14TH STREET-UNION SQUARE	00224603	00011051	00000827	00003026	00000660	00089367	00199841
R046	42ND STREET & GRAND CENTRAL	00207758	00007908	00000323	00001183	00003001	00040759	00096613

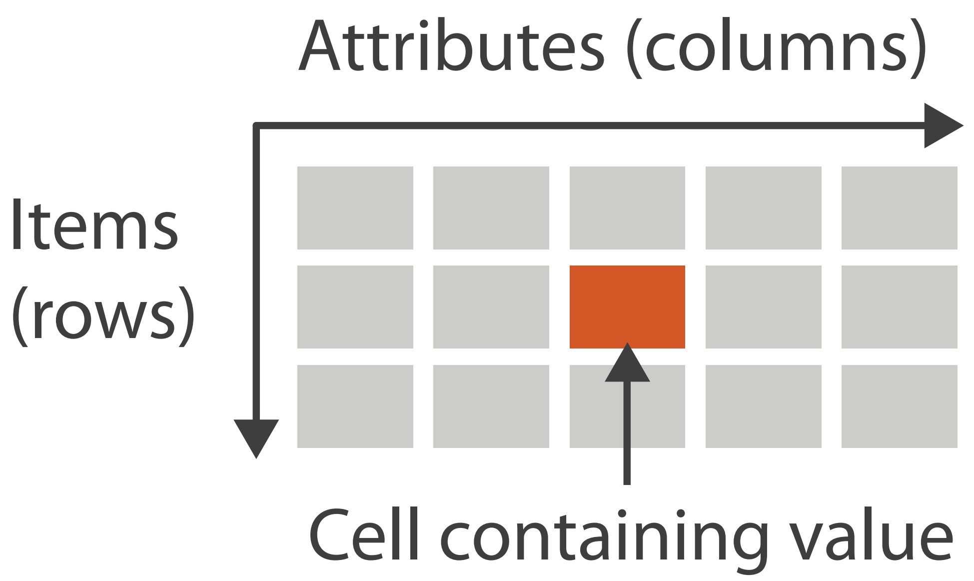
- Semantics: real-world meaning of the data
- Type: structural or mathematical interpretation
- Both often require metadata
  - Sometimes we can infer some of this information
  - Line between data and metadata isn't always clear

Data

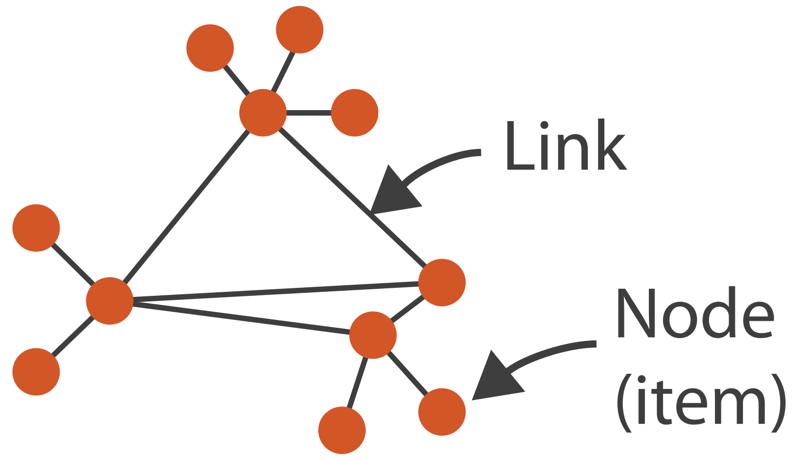
	REMOTE	STATION	FF	SEN/DIS	7-D AFAS UNL	D AFAS/RMF I	JOINT RR TKT	7-D UNL	30-D UNL
1	R011	42ND STREET & 8TH AVENUE	00228985	00008471	00000441	00001455	00000134	00033341	00071255
2	R170	14TH STREET-UNION SQUARE	00224603	00011051	00000827	00003026	00000660	00089367	00199841
3	R046	42ND STREET & GRAND CENTRAL	00207758	00007908	00000323	00001183	00003001	00040759	00096613
4	R012	34TH STREET & 8TH AVENUE	00188311	00006490	00000498	00001279	00003622	00035527	00067483
5	R293	34TH STREET - PENN STATION	00168768	00006155	00000523	00001065	00005031	00030645	00054376
6	R033	42ND STREET/TIMES SQUARE	00159382	00005945	00000378	00001205	00000690	00058931	00078644
7	R022	34TH STREET & 6TH AVENUE	00156008	00006276	00000487	00001543	00000712	00058910	00110466
8	R084	59TH STREET/COLUMBUS CIRCLE	00155262	00009484	00000589	00002071	00000542	00053397	00113966
9	R020	47-50 STREETS/ROCKEFELLER	00143500	00006402	00000384	00001159	00000723	00037978	00090745
10	R179	86TH STREET-LEXINGTON AVE	00142169	00010367	00000470	00001839	00000271	00050328	00125250
11	R023	34TH STREET & 6TH AVENUE	00134052	00005005	00000348	00001112	00000649	00031531	00075040
12	R029	PARK PLACE	00121614	00004311	00000287	00000931	00000792	00025404	00065362
13	R047	42ND STREET & GRAND CENTRAL	00100742	00004273	00000185	00000704	00001241	00022808	00068216
14	R031	34TH STREET & 7TH AVENUE	00095076	00003990	00000232	00000727	00001459	00024284	00038671
15	R017	LEXINGTON AVENUE	00094655	00004688	00000190	00000833	00000754	00020018	00055066
16	R175	8TH AVENUE-14TH STREET	00094313	00003907	00000286	00001144	00000256	00038272	00074661
17	R057	BARCLAYS CENTER	00093804	00004204	00000454	00001386	00001491	00039113	00068119
18	R138	WEST 4TH ST-WASHINGTON SQ	00093562	00004677	00000251	00000965	00000127	00031628	00074458

Dataset Types

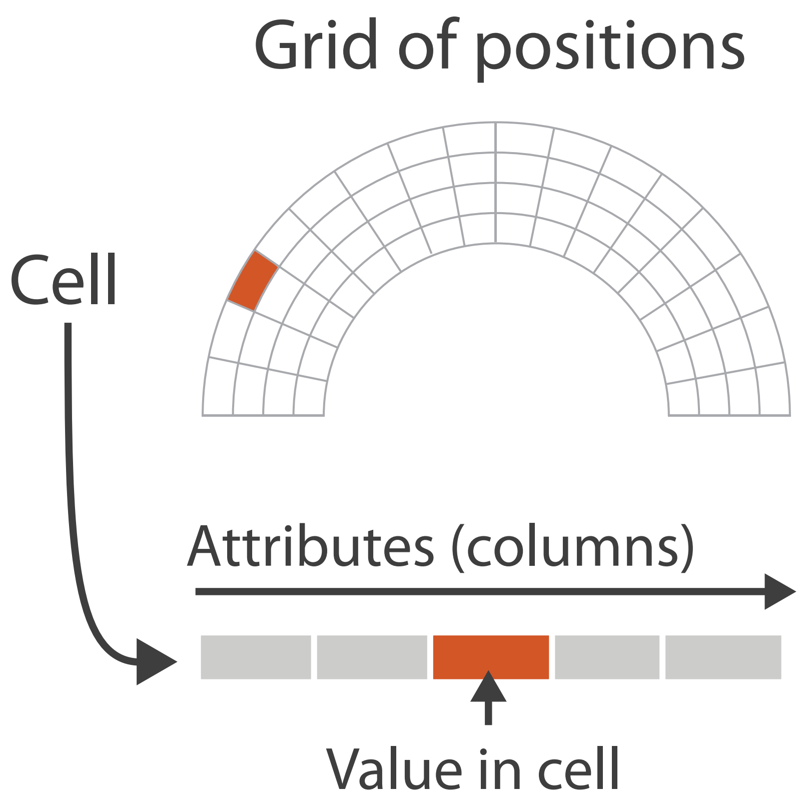
➔ Tables



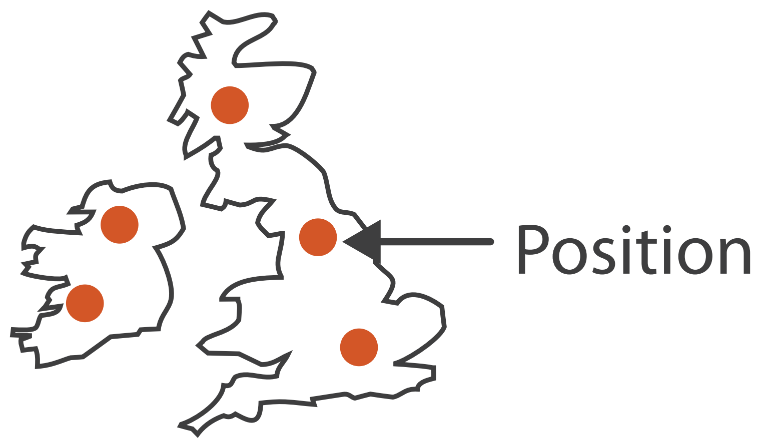
➔ Networks



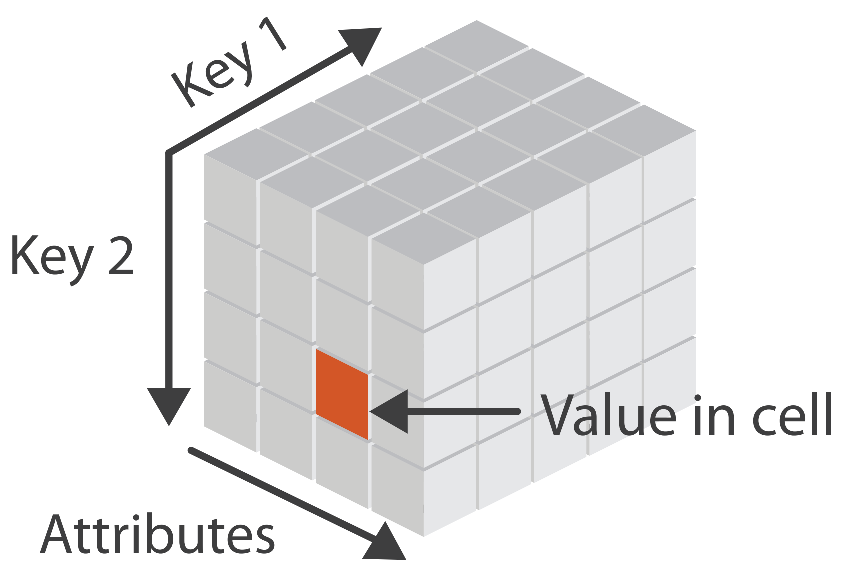
➔ Fields (Continuous)



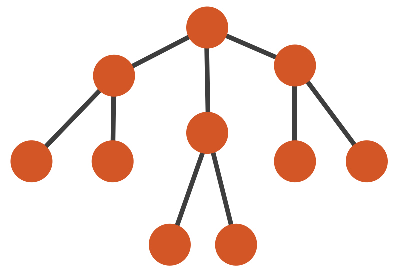
➔ Geometry (Spatial)



➔ *Multidimensional Table*



➔ *Trees*



[Munzner (ill. Maguire), 2014]

Data Terminology

- Items
  - An **item** is an individual discrete entity
  - e.g., a row in a table
- Attributes
  - An **attribute** is some specific property that can be measured, observed, or logged
  - a.k.a. variable, (data) dimension
  - e.g., a column in a table

Tables

A	B	C	S	T	U
Order ID	Order Date	Order Priority	Product Container	Product Base Margin	Ship Date
3	10/14/06	5-Low	Large Box	0.8	10/21/06
6	2/21/08	4-Not Specified	Small Pack	0.55	2/22/08
32	7/16/07	2-High	Small Pack	0.79	7/17/07
32	7/16/07	2-High	Jumbo Box	[illegible]	7/17/07
32	7/16/07	2-High	Medium Box	[illegible]	7/18/07
32	7/16/07	2-High	Medium Box	[illegible]	7/18/07
35	10/23/07	4-Not Specified	Wrap Bag	0.52	10/24/07
35	10/23/07	4-Not Specified	Small Box	0.58	10/25/07
36	11/3/07	1-Urgent	Small Box	0.55	11/3/07
65	3/18/07	1-Urgent	Small Pack	0.49	3/19/07
66	[illegible]	5-Low	Wrap Bag	0.56	1/20/05
69	[illegible]	4-Not Specified	Small Pack	0.44	6/6/05
69	[illegible]	4-Not Specified	Wrap Bag	0.6	6/6/05
70	12/18/06	5-Low	Small Box	0.59	12/23/06
70	12/18/06	5-Low	Wrap Bag	0.82	12/23/06
96	4/17/05	2-High	Small Box	0.55	4/19/05
97	1/29/06	3-Medium	Small Box	0.38	1/30/06
129	11/19/08	5-Low	Small Box	0.37	11/28/08
130	5/8/08	2-High	Small Box	0.37	5/9/08
130	5/8/08	2-High	Medium Box	0.38	5/10/08
130	5/8/08	2-High	Small Box	0.6	5/11/08
132	6/11/06	3-Medium	Medium Box	0.6	6/12/06
132	6/11/06	3-Medium	Jumbo Box	0.69	6/14/06
134	5/1/08	4-Not Specified	Large Box	0.82	5/3/08
135	10/21/07	4-Not Specified	Small Pack	0.64	10/23/07
166	9/12/07	2-High	Small Box	0.55	9/14/07
193	8/8/06	1-Urgent	Medium Box	0.57	8/10/06
194	4/5/08	3-Medium	Wrap Bag	0.42	4/7/08

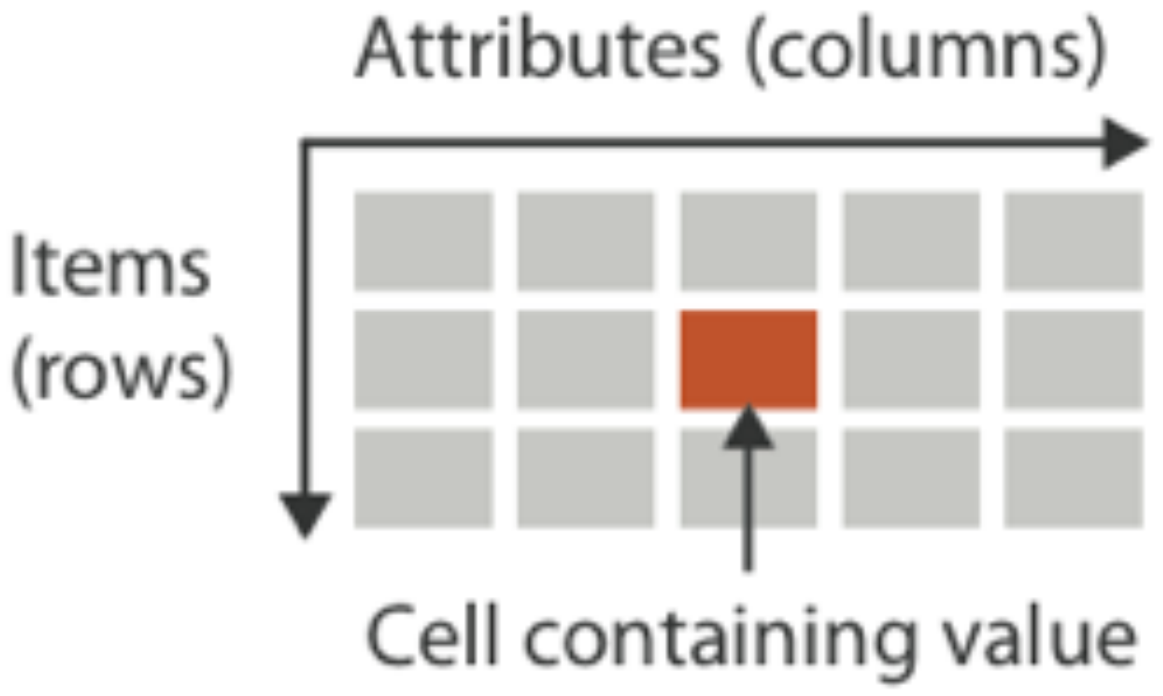
attribute

cell

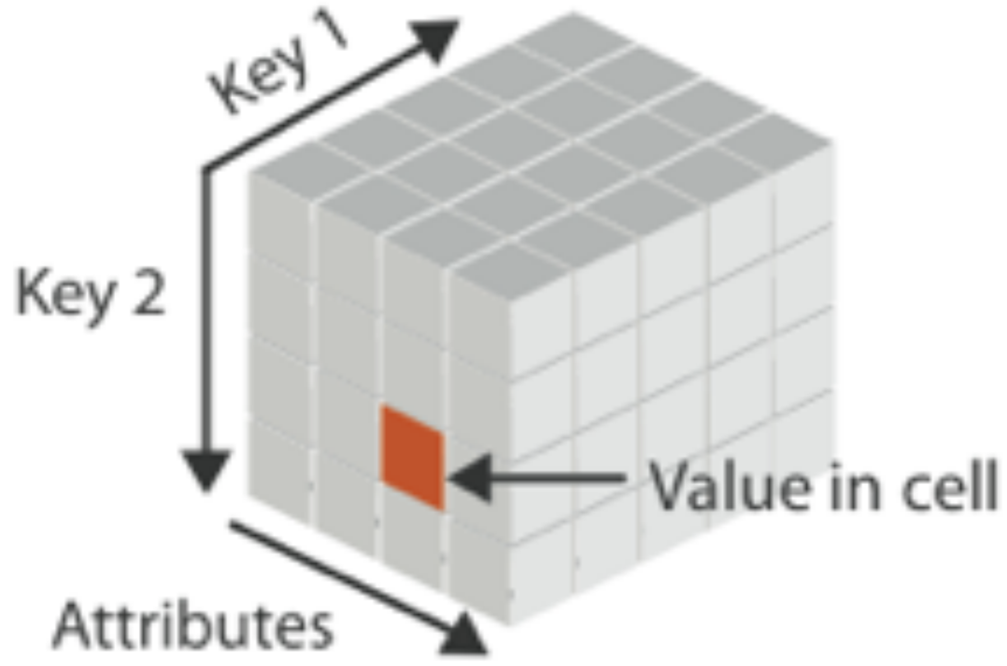
item

Tables

Flat



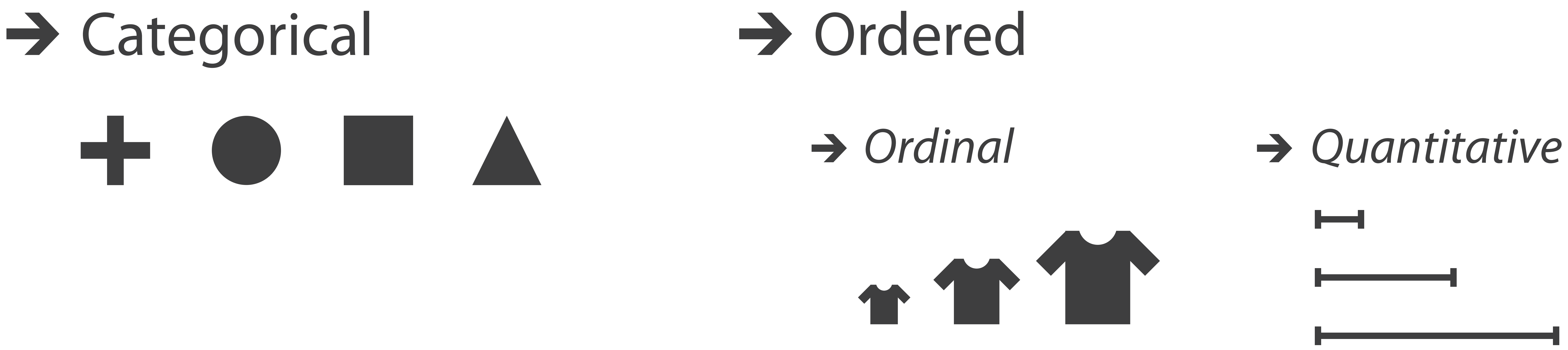
Multidimensional



- Data organized by rows & columns
  - row ~ item (usually)
  - column ~ attribute
  - label ~ attribute name
- Key: identifies each item (row), usually unique
  - Allows **join** of data from 2+ tables
  - Compound key: key split among multiple columns, e.g. (state, year) for population
- Multidimensional:
  - Split compound key
  - e.g. a data cube with (state, year)

[Munzner (ill. Maguire), 2014]

Attribute Types



[Munzner (ill. Maguire), 2014]

Categorial, Ordinal, and Quantitative

A	B	C	S	T	U
Order ID	Order Date	Order Priority	Product Container	Product Base Margin	Ship Date
3	10/14/06	5-Low	Large Box	0.8	10/21/06
6	2/21/08	4-Not Specified	Small Pack	0.55	2/22/08
32	7/16/07	2-High	Small Pack	0.79	7/17/07
32	7/16/07	2-High	Jumbo Box	0.72	7/17/07
32	7/16/07	2-High	Medium Box	0.6	7/18/07
32	7/16/07	2-High	Medium Box	0.65	7/18/07
35	10/23/07	4-Not Specified	Wrap Bag	0.52	10/24/07
35	10/23/07	4-Not Specified	Small Box	0.58	10/25/07
36	11/3/07	1-Urgent	Small Box	0.55	11/3/07
65	3/18/07	1-Urgent	Small Pack	0.49	3/19/07
66	1/20/05	5-Low	Wrap Bag	0.56	1/20/05
69	6/4/05	4-Not Specified	Small Pack	0.44	6/6/05
69	6/4/05	4-Not Spec		0.6	6/6/05
70	12/18/06	5-Low		0.59	12/23/06
70	12/18/06	5-Low		0.82	12/23/06
96	4/17/05	2-High		0.55	4/19/05
97	1/29/06	3-Medium		0.38	1/30/06
129	11/19/08	5-Low		0.37	11/28/08
130	5/8/08	2-High	Small Box	0.37	5/9/08
130	5/8/08	2-High	Medium Box	0.38	5/10/08
130	5/8/08	2-High	Small Box	0.6	5/11/08
132	6/11/06	3-Medium	Medium Box	0.6	6/12/06
132	6/11/06	3-Medium	Jumbo Box	0.69	6/14/06
134	5/1/08	4-Not Specified	Large Box	0.82	5/3/08
135	10/21/07	4-Not Specified	Small Pack	0.64	10/23/07
166	9/12/07	2-High	Small Box	0.55	9/14/07
193	8/8/06	1-Urgent	Medium Box	0.57	8/10/06
194	4/5/08	3-Medium	Wrap Bag	0.42	4/7/08

quantitative
ordinal
categorical

Categorial, Ordinal, and Quantitative

A	B	C	S	T	U
Order ID	Order Date	Order Priority	Product Container	Product Base Margin	Ship Date
3	10/14/06	5-Low	Large Box	0.8	10/21/06
6	2/21/08	4-Not Specified	Small Pack	0.55	2/22/08
32	7/16/07	2-High	Small Pack	0.79	7/17/07
32	7/16/07	2-High	Jumbo Box	0.72	7/17/07
32	7/16/07	2-High	Medium Box	0.6	7/18/07
32	7/16/07	2-High	Medium Box	0.65	7/18/07
35	10/23/07	4-Not Specified	Wrap Bag	0.52	10/24/07
35	10/23/07	4-Not Specified	Small Box	0.58	10/25/07
36	11/3/07	1-Urgent	Small Box	0.55	11/3/07
65	3/18/07	1-Urgent	Small Pack	0.49	3/19/07
66	1/20/05	5-Low	Wrap Bag	0.56	1/20/05
69	6/4/05	4-Not Specified	Small Pack	0.44	6/6/05
69	6/4/05	4-Not Spec		0.6	6/6/05
70	12/18/06	5-Low		0.59	12/23/06
70	12/18/06	5-Low		0.82	12/23/06
96	4/17/05	2-High		0.55	4/19/05
97	1/29/06	3-Medium		0.38	1/30/06
129	11/19/08	5-Low		0.37	11/28/08
130	5/8/08	2-High	Small Box	0.37	5/9/08
130	5/8/08	2-High	Medium Box	0.38	5/10/08
130	5/8/08	2-High	Small Box	0.6	5/11/08
132	6/11/06	3-Medium	Medium Box	0.6	6/12/06
132	6/11/06	3-Medium	Jumbo Box	0.69	6/14/06
134	5/1/08	4-Not Specified	Large Box	0.82	5/3/08
135	10/21/07	4-Not Specified	Small Pack	0.64	10/23/07
166	9/12/07	2-High	Small Box	0.55	9/14/07
193	8/8/06	1-Urgent	Medium Box	0.57	8/10/06
194	4/5/08	3-Medium	Wrap Bag	0.42	4/7/08

quantitative
ordinal
categorical

Attribute Types

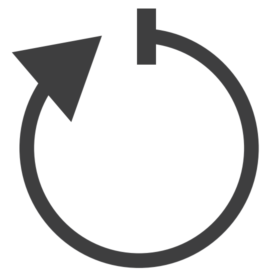
- May be further specified for computational storage/processing
  - **Categorical**: string, boolean, blood type
  - **Ordered**: enumeration, t-shirt size
  - **Quantitative**: integer, float, fixed decimal, datetime
- Sometimes, types can be **inferred** from the data
  - e.g. numbers and none have decimal points → integer
  - could be incorrect (data doesn't have floats, but could be)

Ordering Direction

- ➔ Sequential

- ➔ Diverging

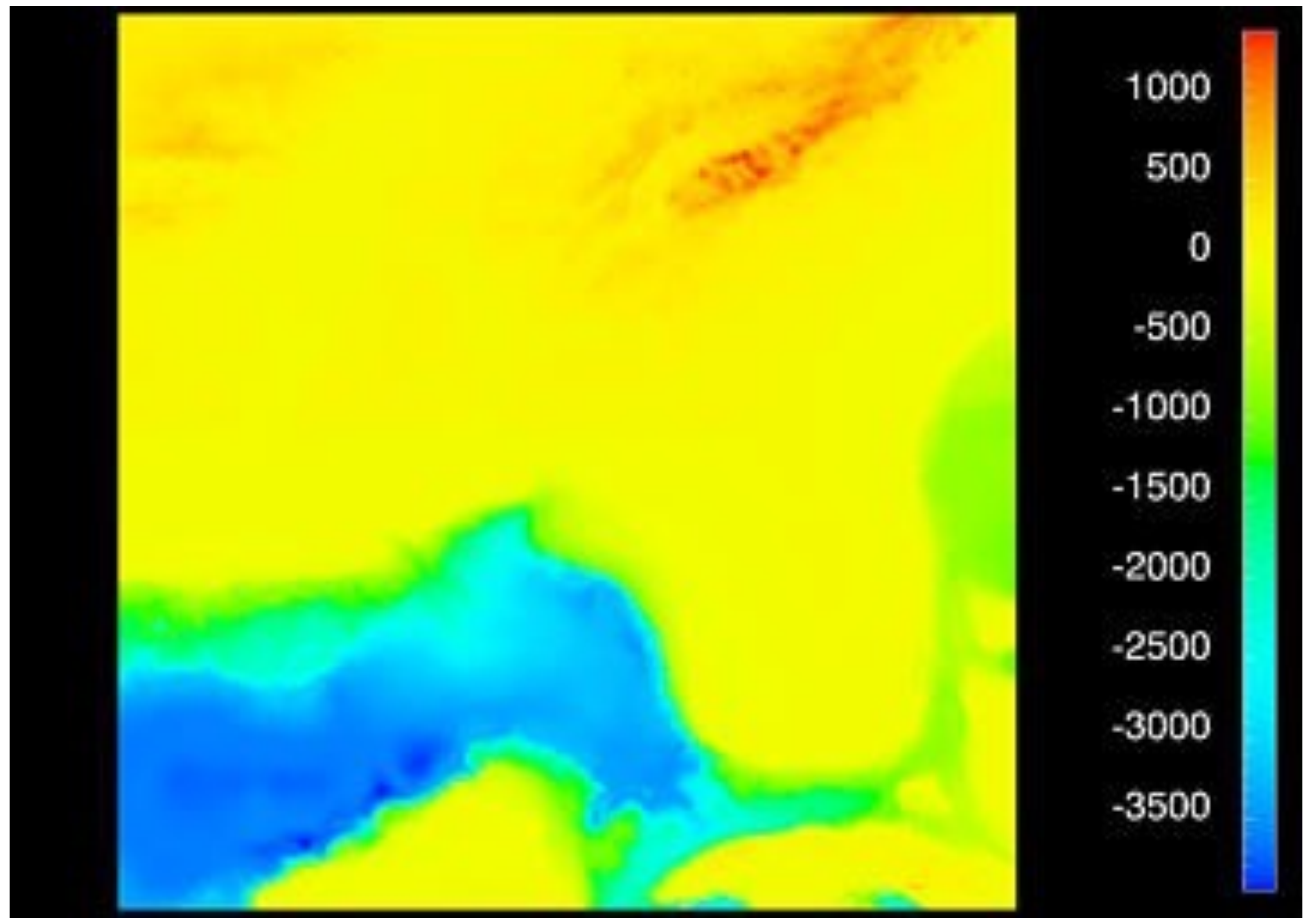
- ➔ Cyclic



[Munzner (ill. Maguire), 2014]

Sequential and Diverging Data

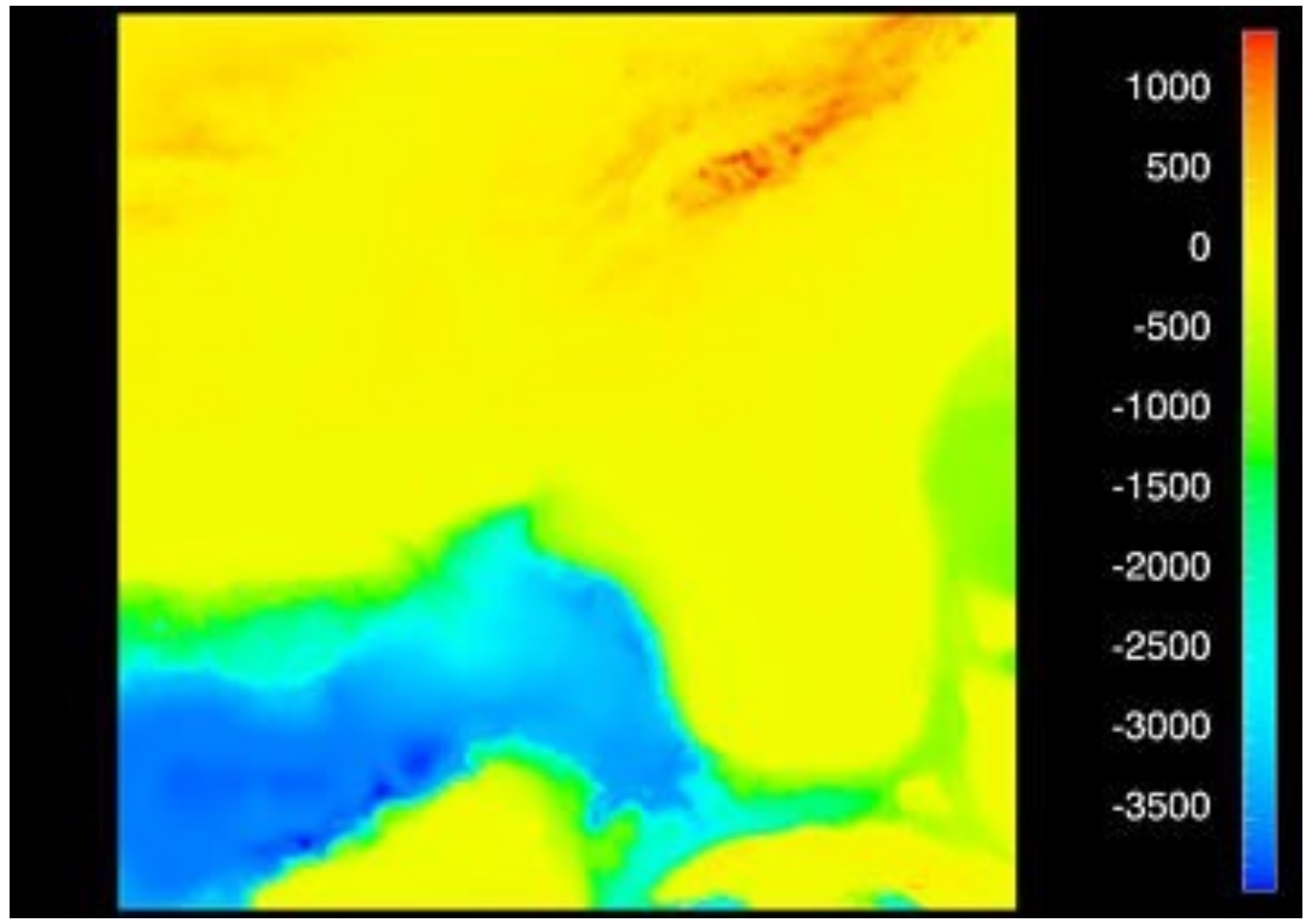
- Sequential: homogenous range from a minimum to a maximum
  - Examples: Land elevations, ocean depths
- Diverging: can be deconstructed into two sequences pointing in opposite directions
  - Has a **zero point** (not necessary 0)
  - Example: Map of both land elevation and ocean depth

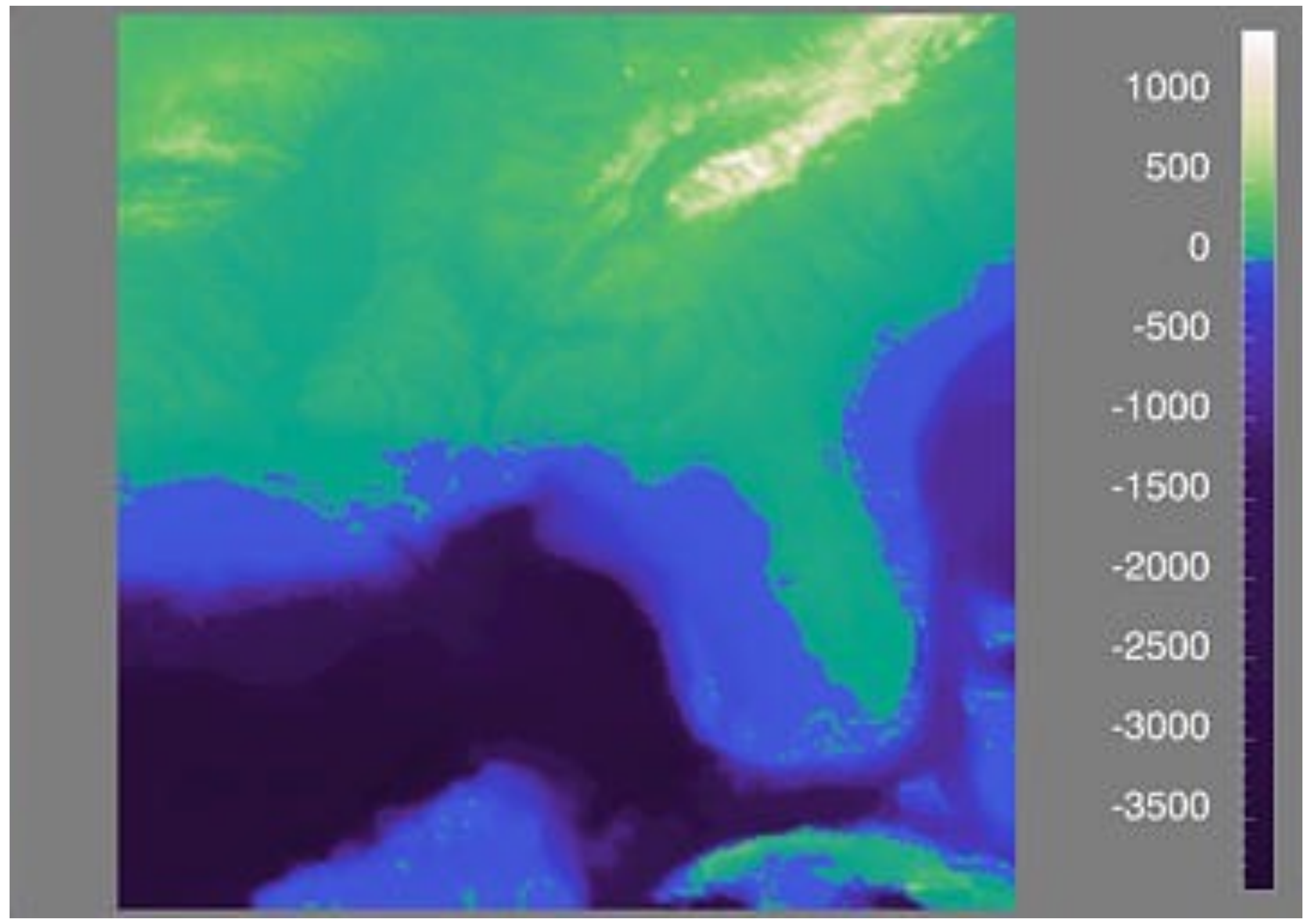


[Rogowitz & Treinish, 1998]

Sequential and Diverging Data

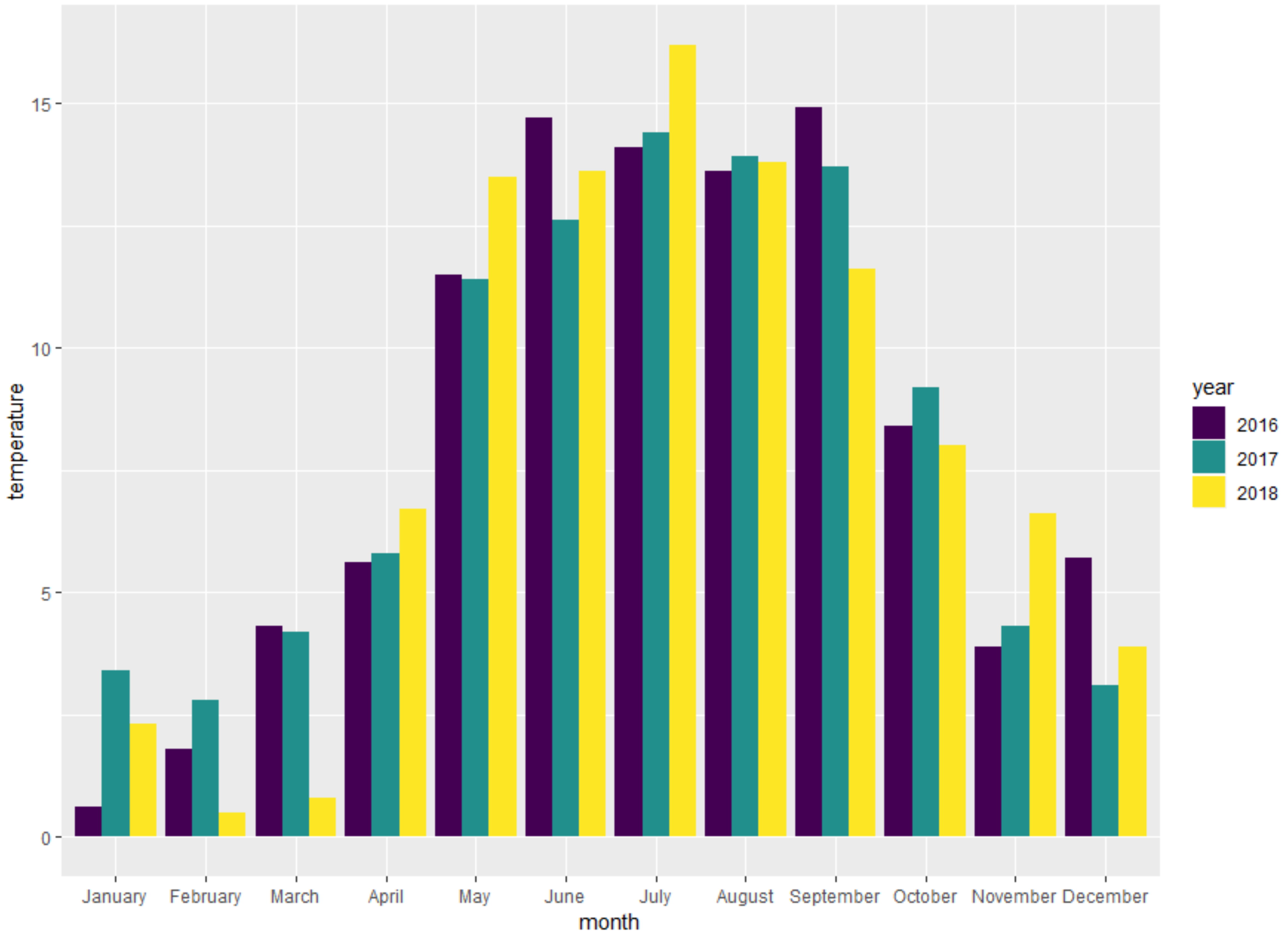
- Sequential: homogenous range from a minimum to a maximum
  - Examples: Land elevations, ocean depths
- Diverging: can be deconstructed into two sequences pointing in opposite directions
  - Has a **zero point** (not necessary 0)
  - Example: Map of both land elevation and ocean depth





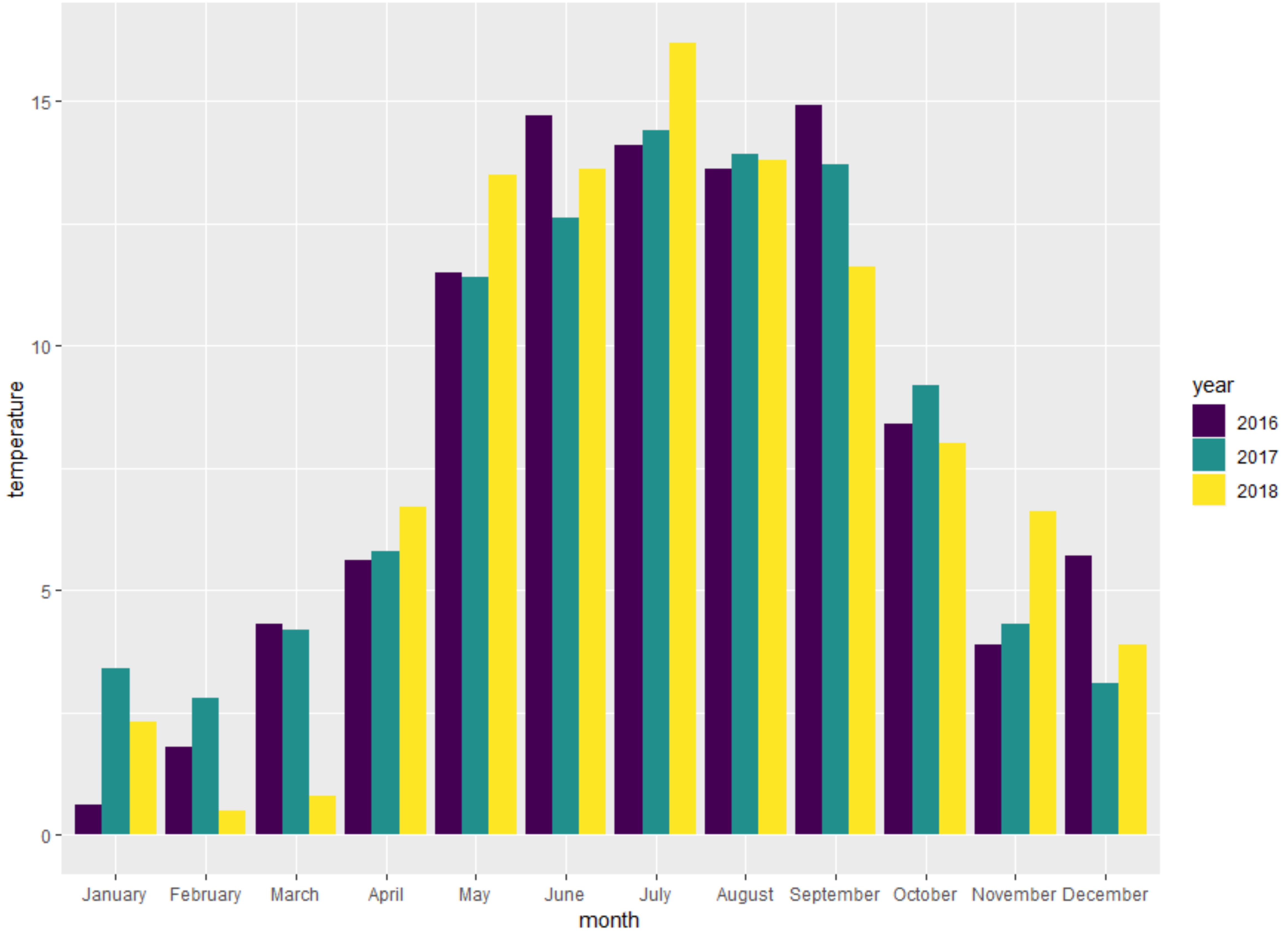
[Rogowitz & Treinish, 1998]

Cyclic Data

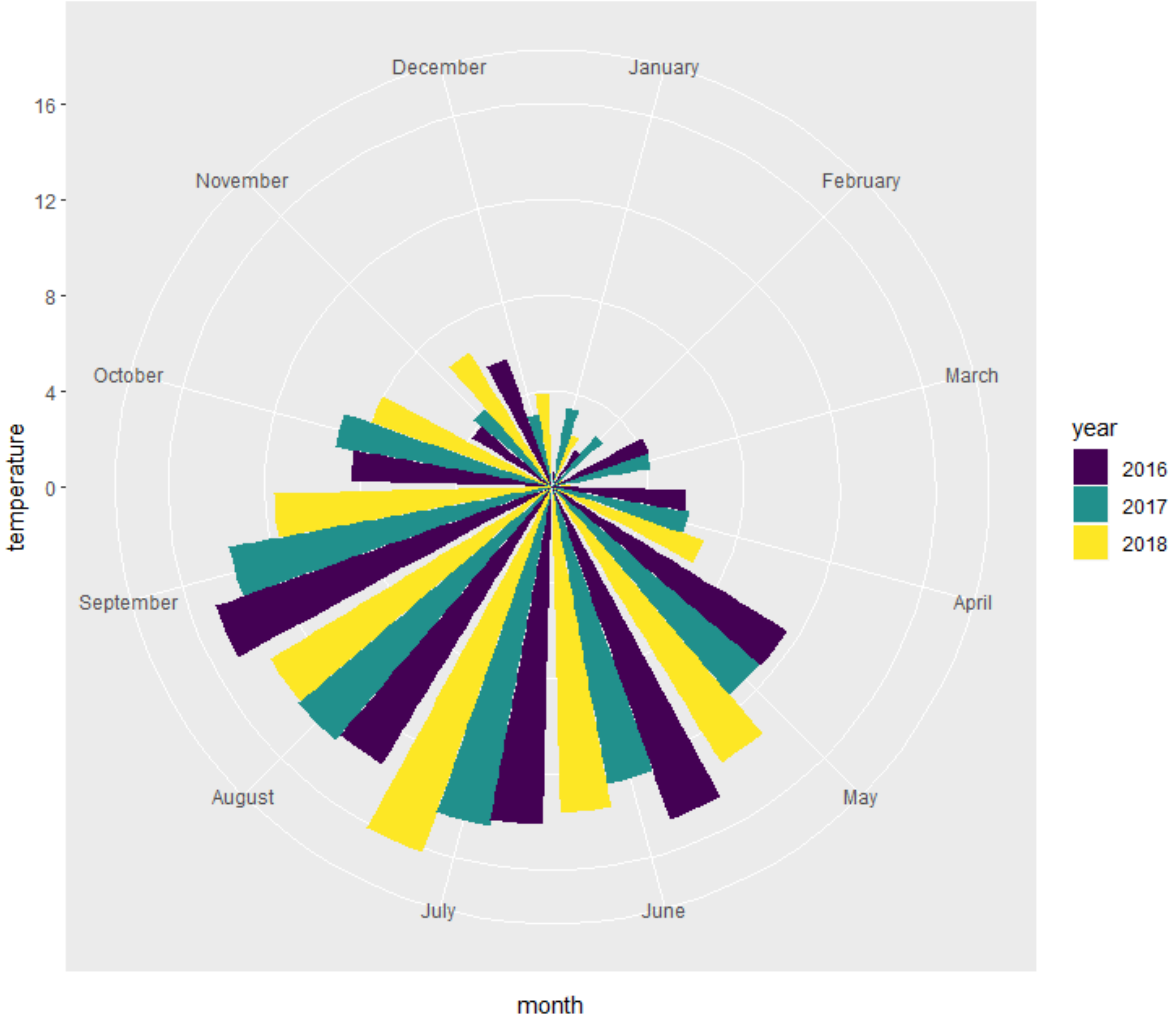


15
10
5
0
temperature
January
February
March
April
May
June
July
August
September
October
November
December
month
year
2016
2017
2018

Cyclic Data



0
5
10
15
temperature
January
February
March
April
May
June
July
August
September
October
November
December
month
year
2016
2017
2018



16
12
8
4
0
temperature
December
January
February
March
April
May
June
July
August
September
October
November
month
year
2016
2017
2018

Semantics

- The meaning of the data
- Example: 94023, 90210, 02747, 60115

Semantics

- The meaning of the data
- Example: 94023, 90210, 02747, 60115
  - Attendance at college football games?

Semantics

- The meaning of the data
- Example: 94023, 90210, 02747, 60115
  - Attendance at college football games?
  - Salaries?

Semantics

- The meaning of the data
- Example: 94023, 90210, 02747, 60115
  - Attendance at college football games?
  - Salaries?
  - Zip codes?
- Cannot always infer based on what the data looks like
- Often require semantics to better understand data, column names help
- May also include rules about data: a zip code is part of an address that uniquely identifies a residence
- Useful for asking good questions about the data

Data Model vs. Conceptual Model

- Data Model: raw data that has a specific data type (e.g. floats):
  - Temperature Example: `[32.5, 54.0, -17.3]` (floats)
- Conceptual Model: how we think about the data
  - Includes semantics, reasoning
  - Temperature Example:
    - Quantitative: `[32.50, 54.00, -17.30]`

[via A. Lex, 2015]

Data Model vs. Conceptual Model

- Data Model: raw data that has a specific data type (e.g. floats):
  - Temperature Example: `[32.5, 54.0, -17.3]` (floats)
- Conceptual Model: how we think about the data
  - Includes semantics, reasoning
  - Temperature Example:
    - Quantitative: `[32.50, 54.00, -17.30]`
    - Ordered: `[warm, hot, cold]`

[via A. Lex, 2015]

Data Model vs. Conceptual Model

- Data Model: raw data that has a specific data type (e.g. floats):
  - Temperature Example: `[32.5, 54.0, -17.3]` (floats)
- Conceptual Model: how we think about the data
  - Includes semantics, reasoning
  - Temperature Example:
    - Quantitative: `[32.50, 54.00, -17.30]`
    - Ordered: `[warm, hot, cold]`
    - Categorical: `[not burned, burned, not burned]`

[via A. Lex, 2015]

Derived Data

Derived Data

- Often, data in its original form isn't as useful as we would like
- Examples: Data about a basketball team's games

Derived Data

- Often, data in its original form isn't as useful as we would like
- Examples: Data about a basketball team's games
- Example 1: `1stHalfPoints, 2ndHalfPoints`
  - More useful to know total number of points
  - `Points = 1stHalfPoints + 2ndHalfPoints`

Derived Data

- Often, data in its original form isn't as useful as we would like
- Examples: Data about a basketball team's games
- Example 1: `1stHalfPoints, 2ndHalfPoints`
  - More useful to know total number of points
  - `Points = 1stHalfPoints + 2ndHalfPoints`
- Example 2: `Points, OpponentPoints`
  - Want to have a column indicating win/loss
  - `Win = True if (Points > OpponentPoints) else False`

Derived Data

- Often, data in its original form isn't as useful as we would like
- Examples: Data about a basketball team's games
- Example 1: `1stHalfPoints, 2ndHalfPoints`
  - More useful to know total number of points
  - `Points = 1stHalfPoints + 2ndHalfPoints`
- Example 2: `Points, OpponentPoints`
  - Want to have a column indicating win/loss
  - `Win = True if (Points > OpponentPoints) else False`
- Example 3: `Points`
  - Want to have a column indicating how that point total ranks
  - `Rank = index in sorted list of all Point values`

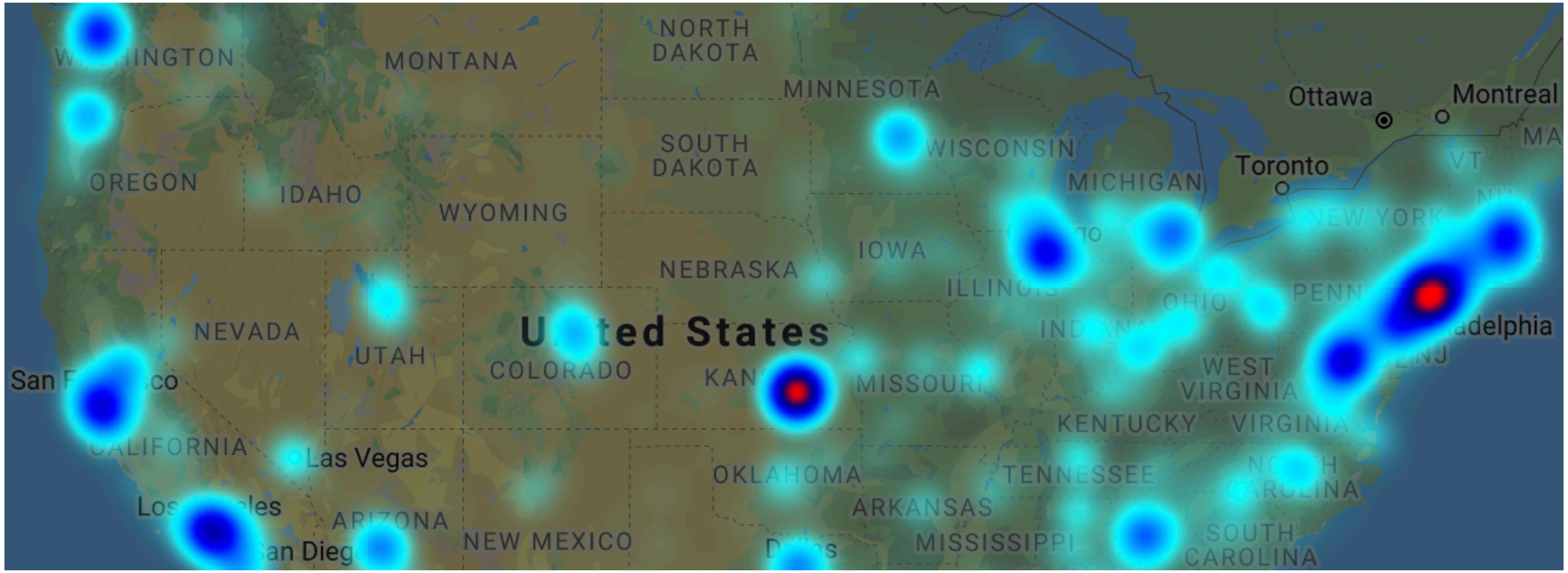
What if data isn't correct/trustworthy/in the right format?

Dirty Data

[Flickr]

Geolocation Errors

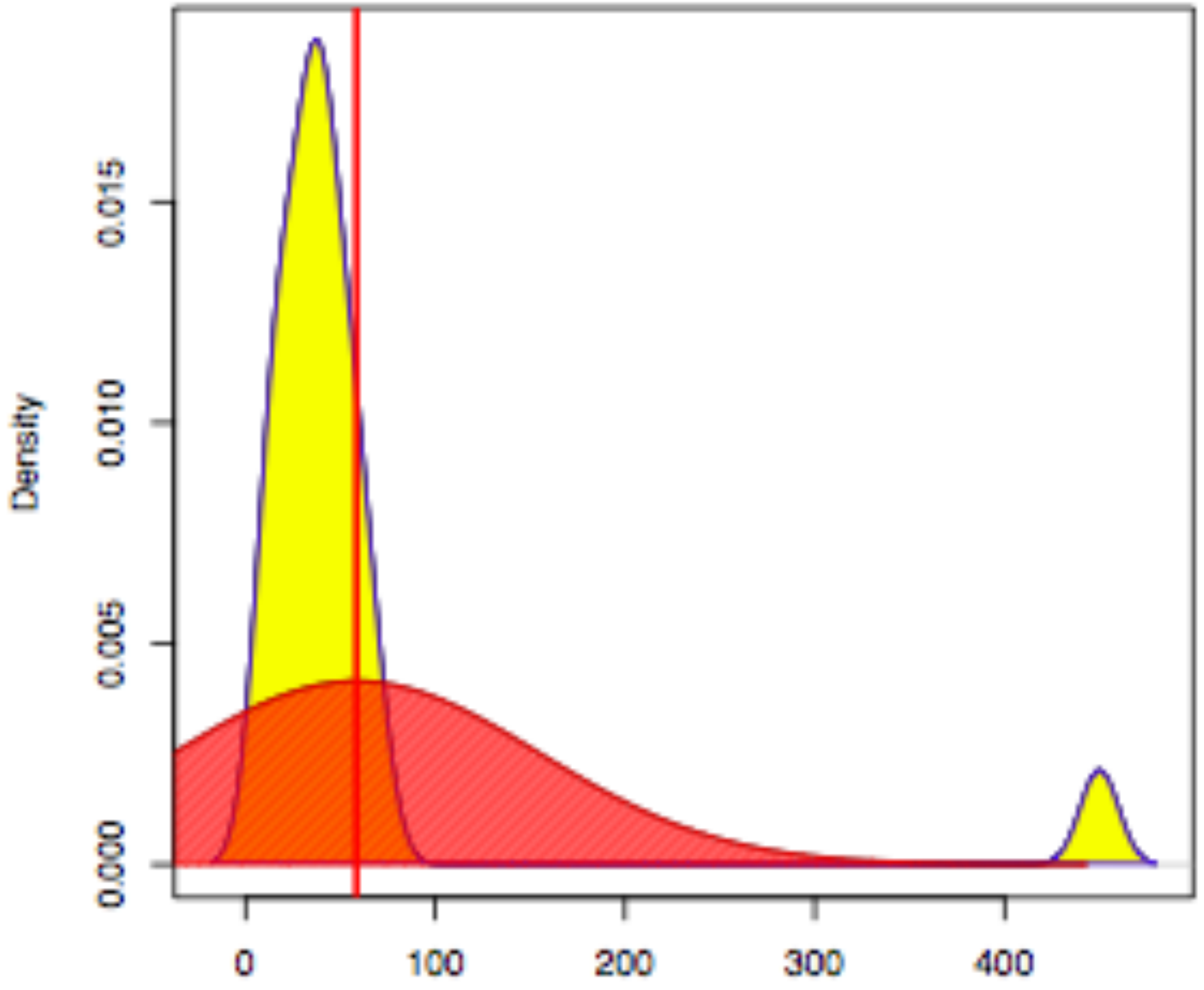
- Maxmind helps companies determine where users are located based on IP address
- "How a quiet Kansas home wound up with 600 million IP addresses and a world of trouble" [Washington Post, 2016]



Numeric Outliers

12	13	14	21	22	26	33	35	36	37	39	42	45	47	54	57	61	68	450

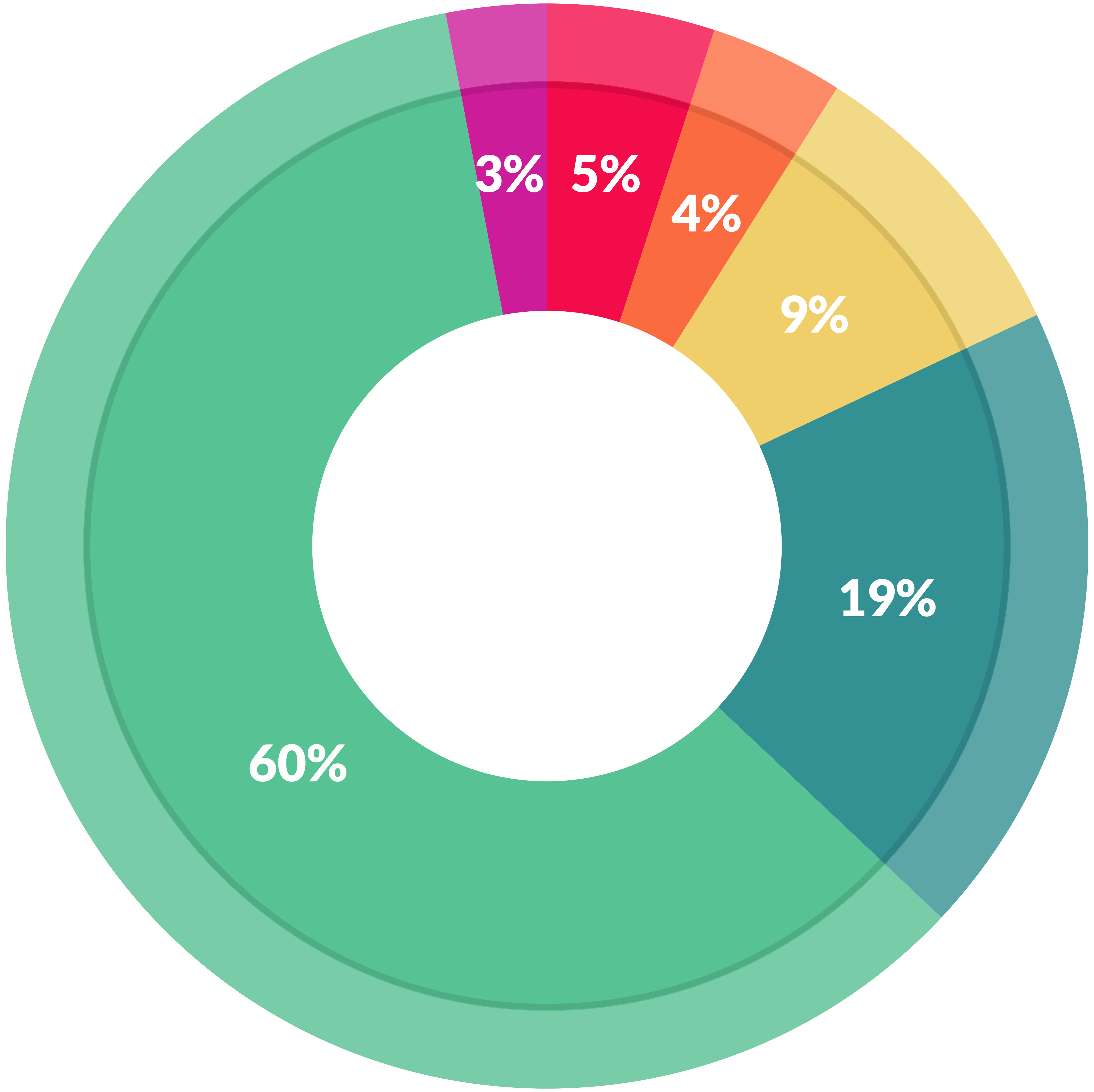
ages of employees (US)



- median 37
- mean 58.52632
- variance 9252.041

[J. Hellerstein via J. Canny et al.]

This takes a lot of time!

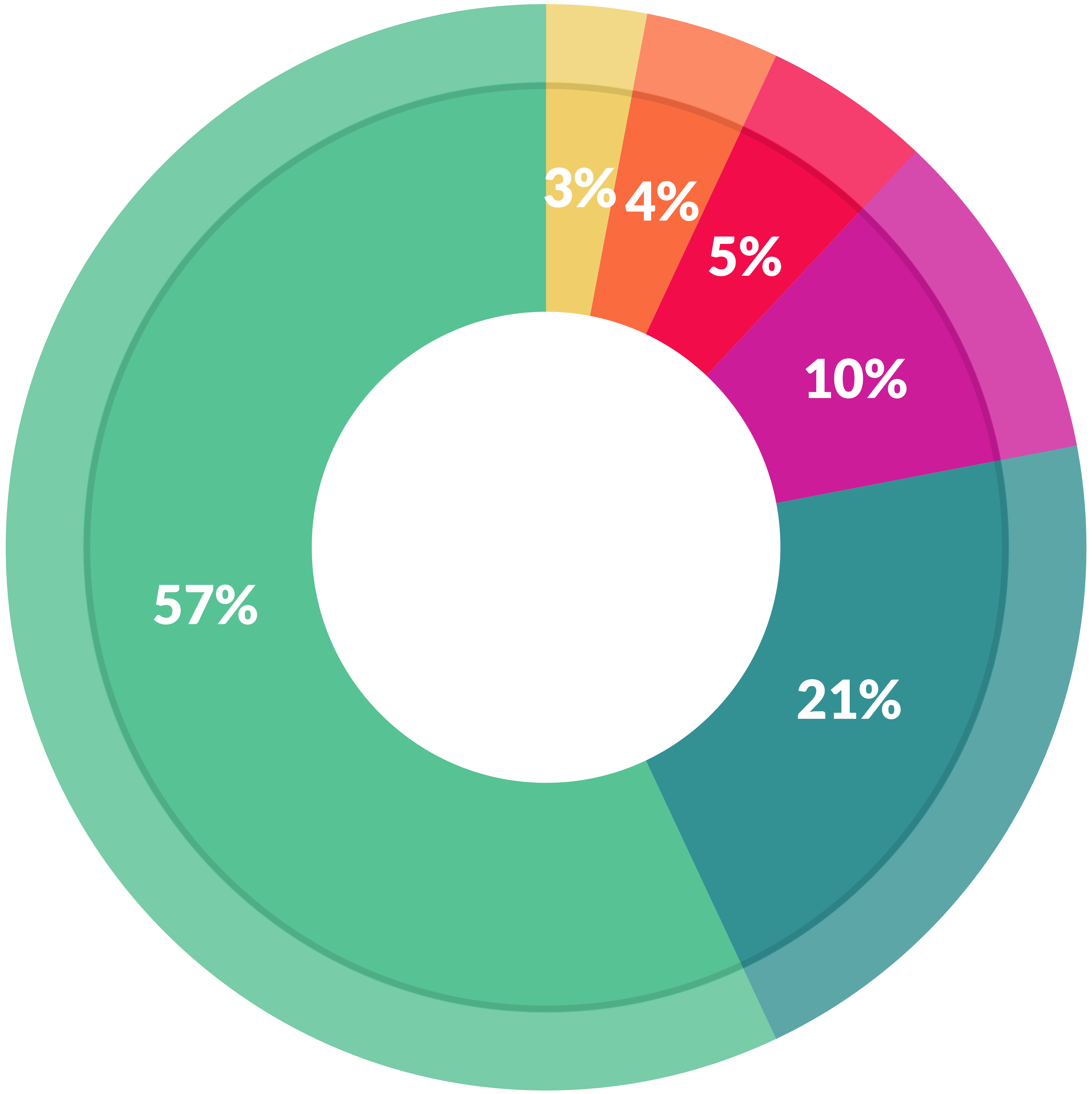


What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

[CrowdFlower Data Science Report, 2016]

…and it isn't the most fun thing to do



What's the least enjoyable part of data science?

- *Building training sets: 10%*
- *Cleaning and organizing data: 57%*
- *Collecting data sets: 21%*
- *Mining data for patterns: 3%*
- *Refining algorithms: 4%*
- *Other: 5%*

[CrowdFlower Data Science Report, 2016]

Dirty Data: Statistician's View

- Some process produces the data
- Want a model but have non-ideal samples:
  - Distortion: some samples corrupted by a process
  - Selection bias: likelihood of a sample depends on its value
  - Left and right censorship: users come and go from scrutiny
  - Dependence: samples are not independent (e.g. social networks)
- You can add/augment models for different problems, but cannot model everything
- Trade-off between accuracy and simplicity

[J. Canny et al.]

Dirty Data: Database Expert's View

- Got a dataset
- Some values are missing, corrupted, wrong, duplicated
- Results are absolute (relational model)
- Better answers come from improving the quality of values in the dataset

[J. Canny et al.]

Dirty Data: Domain Expert's View

- Data doesn't look right
- Answer doesn't look right
- What happened?
- Domain experts carry an implicit model of the data they test against
- You don't always need to be a domain expert to do this
  - Can a person run 50 miles an hour?
  - Can a mountain on Earth be 50,000 feet above sea level?
  - Use common sense

[J. Canny et al.]

Dirty Data: Data Scientist's View

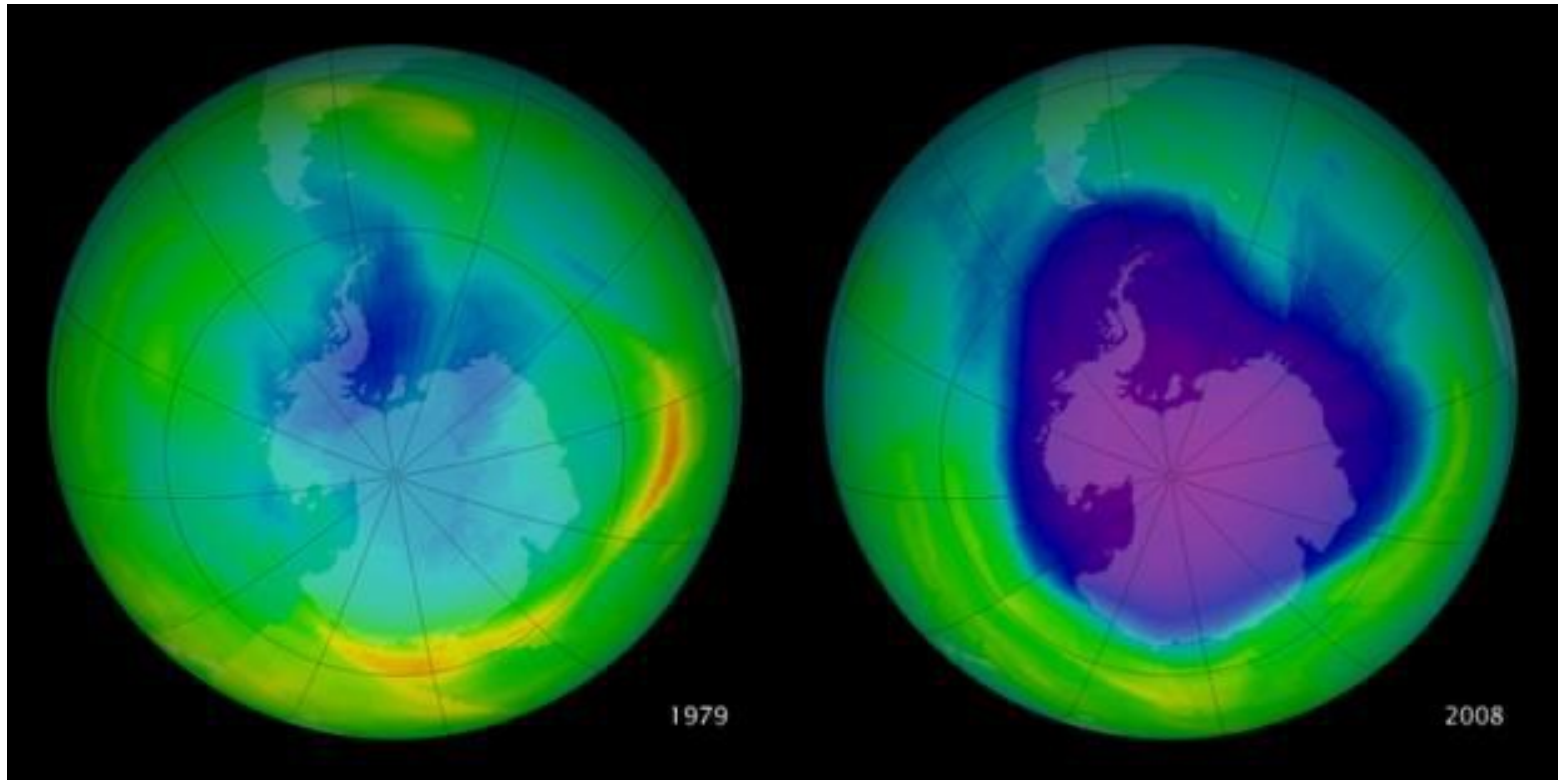
- Combination of the previous three views
- All of the views present problems with the data
- The goal may dictate the solutions:
  - Median value: don't worry too much about crazy outliers
  - Generally, aggregation is less susceptible by numeric errors
  - Be careful, the data may be correct…

[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]

Where does dirty data originate?

- Source data is bad, e.g. person entered it incorrectly
- Transformations corrupt the data, e.g. certain values processed incorrectly due to a software bug
- Integration of different datasets causes problems
- Error propagation: one error is magnified

[J. Canny et al.]

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.]

Data Wrangling

- Data wrangling: transform raw data to a more meaningful format that can be better analyzed
- Data cleaning: getting rid of inaccurate data
- Data transformations: changing the data from one representation to another
- Data reshaping: reorganizing the data
- Data merging: combining two datasets

Data Cleaning

Wrangler: Interactive Visual Specification of Data Transformation Scripts

S. Kandel, A. Paepcke, J. Hellerstein, J. Heer

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"

Your Critique/Questions

Example Critique

- Summary: Wrangler tackles data wrangling tasks by combining a language for specifying operations with an interface allowing users to specify the types of changes they are interested; the system can then generate suggested operations and demonstrates them on demand
- Critique: The suggestions may lead to states that a user cannot recover from easily. Suppose a suggestion looks like it works well, but a user later realizes was incorrect. They can backtrack, but it's often unclear where to and which other path to take. In addition, a user has to have some idea of the constructs of the language in order to edit parameters. Without a good idea of the impact of the parameters, the work may become as tedious as manual correction. Perhaps a more example-based strategy could help.

Previous Work: Potter's Wheel

- V. Raman and J. Hellerstein, 2001
- Defines structure extractions for identifying fields
- Defines transformations on the data
- Allows user interaction

Potter's Wheel: Structure Extraction

Example Column Value (Example erroneous values)	# Structures Enumerated	Final Structure Chosen (Punc = Punctuation)
-60	5	*Integer*
UNITED, DELTA, AMERICAN etc.	5	*IspellWord*
SFO, LAX etc. (JFK to OAK)	12	*AllCapsWord*
1998/01/12	9	*Int Punc(/) Int Punc(/) Int*
M, Tu, Thu etc.	5	*Capitalized Word*
06:22	5	*Int(len 2) Punc(:) Int(len 2)*
12.8.15.147 (ferret03.webtop.com)	9	*Double Punc('.') Double*
"GET\b (\b)	5	*Punc(") IspellWord Punc(\)*
/postmodern/lecs/xia/sld013.htm	4	ξ^*
HTTP	3	*AllCapsWord(HTTP)*
/1.0	6	*Punc(/) Double(1.0)*

[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, 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]

Potter's Wheel: Example

		Stewart,Bob
Anna	Davis	
		Dole,Jerry
Joan	Marsh	

Format
'(.*), (.*)' to '\2 \1'

		Bob Stewart
Anna	Davis	
		Jerry Dole
Joan	Marsh	

Split at ' '

		Bob	Stewart
Anna	Davis		
		Jerry	Dole
Joan	Marsh		

2 Merges

Bob	Stewart
Anna	Davis
Jerry	Dole
Joan	Marsh

[V. Raman and J. Hellerstein, 2001]

Potter's Wheel: Inferring Structure from Examples

<table>
<tr><th>Example Values Split By User
(| is user specified split position)</th><th>Inferred Structure</th><th>Comments</th></tr>
<tr><td>Taylor, Jane |, $52,072
Blair, John |, $73,238
Tony Smith |, $1,00,533</td><td>$(< \xi^* > < \text{','}\ \mathit{Money} >)$</td><td>Parsing is doable despite no good delimiter. A regular expression domain can infer a structure of $[0-9,]* for last component.</td></tr>
<tr><td>MAA |to| SIN
JFK |to| SFO
LAX |–| ORD
SEA |/| OAK</td><td>$(<\mathit{len\ 3\ identifier}> < \xi^* > < \mathit{len\ 3\ identifier}>)$</td><td>Parsing is possible despite multiple delimiters.</td></tr>
<tr><td>321 Blake #7 |, Berkeley |, CA 94720
719 MLK Road |, Fremont |, CA 95743</td><td>$(<\mathit{number}\ \xi^* > < \text{','}\ \mathit{word}> <\text{','}\ \mathit{(2\ letter\ word)\ (5\ letter\ integer)}>)$</td><td>Parsing is easy because of consistent delimiter.</td></tr>
</table>

[V. Raman and J. Hellerstein, 2001]