

Handling Data

Manoel Horta Ribeiro
manoel@cs.princeton.edu

Announcements

- **This Thursday:** Mock quiz.
- **This Friday:** First precept & homework.
- Office hours are now properly listed on Canvas!

Getting Data



Organizing Data



Wrangling Data



Getting Data



Organizing Data



Wrangling Data



Suppose we have the data...

- The way we format and represent this data *matters a lot!*
- Imagine you save all your data in a gigantic JSON file
 - *How long does it take to find everyone who lives in zip code 08540?*

Id	customer_id	type	street	Zip
1	CUST-9912	home	14 Witherspoon St	_08540
2	CUST-9912	work	1 Nassau Hall	_08544
3	CUST-9875	home	40 Chestnut St	_08540

```
{
  "order_id": "ORD-2026-004281",
  "created_at": "2026-04-10T11:42:13Z",
  "status": "processing",
  "customer": {
    "customer_id": "CUST-9912",
    "name": "Ana Martins",
    "email": "ana.martins@example.com",
    "phone": "+1-609-555-0182",
    "addresses": [
      {
        "type": "home",
        "street": "14 Witherspoon St",
        "city": "Princeton",
        "state": "NJ",
        "zip": "08542",
        "country": "USA"
      },
      {
        "type": "work",
        "street": "1 Nassau Hall",
        "city": "Princeton",
        "state": "NJ",
        "zip": "08544",
        "country": "USA"
      }
    ]
  }
}
```

Flat data model

- Each record is independent and has the same attributes
- *Examples:* a web server's log file, .csv files
- Very convenient and flexible (And ineffective)

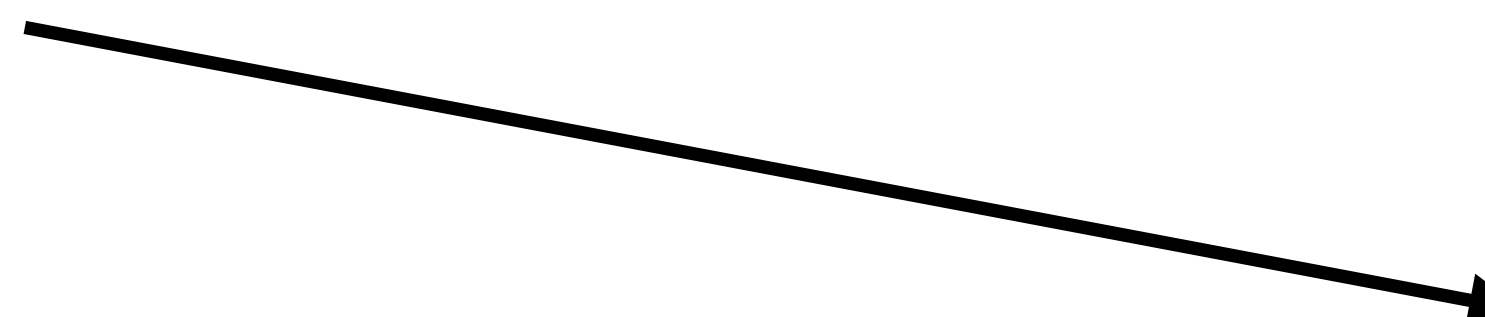
```
[11/Dec/2018:11:01:28 -0600] "GET /blog/seo/robots-txt-mistakes-best-uses-guidelines/ HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /wp-content/themes/portent_portent/assets/s... HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /wp-content/themes/portent_portent/assets/s... HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /about/careers HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /blog/internet-marketing/the_internet_marketing_industry HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /blog/featured HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /services/ppc HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /services/amazon HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /case-study/expand-audience-reach-content-strategy HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /wp-content/themes/portent_portent/assets/s... HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /blog/seo/structure-site-navigation-for-seo HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /blog/author/caleb-cosper HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /about HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /contact HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /case-study/ppc-campaign-restructure-fuels-growth HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /about/philosophy HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /services/seo HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /services/smb HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /wp-content/themes/portent_portent/assets/s... HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /wp-content/cache/autoptimize/css/autoptimize.css HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /resources HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /tools HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /wp-content/cache/autoptimize/css/autoptimize.css HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
[11/Dec/2018:11:01:28 -0600] "GET /blog/seo HTTP/1.1" 502 182 "-" "Screaming Frog SEO Spider"
```

Relational data model

- Various types of entities are connected by relationships
- Tables represent...
 - *entities*
 - *relationships*
- The model behind most databases (PostgreSQL, SQLITE, etc)



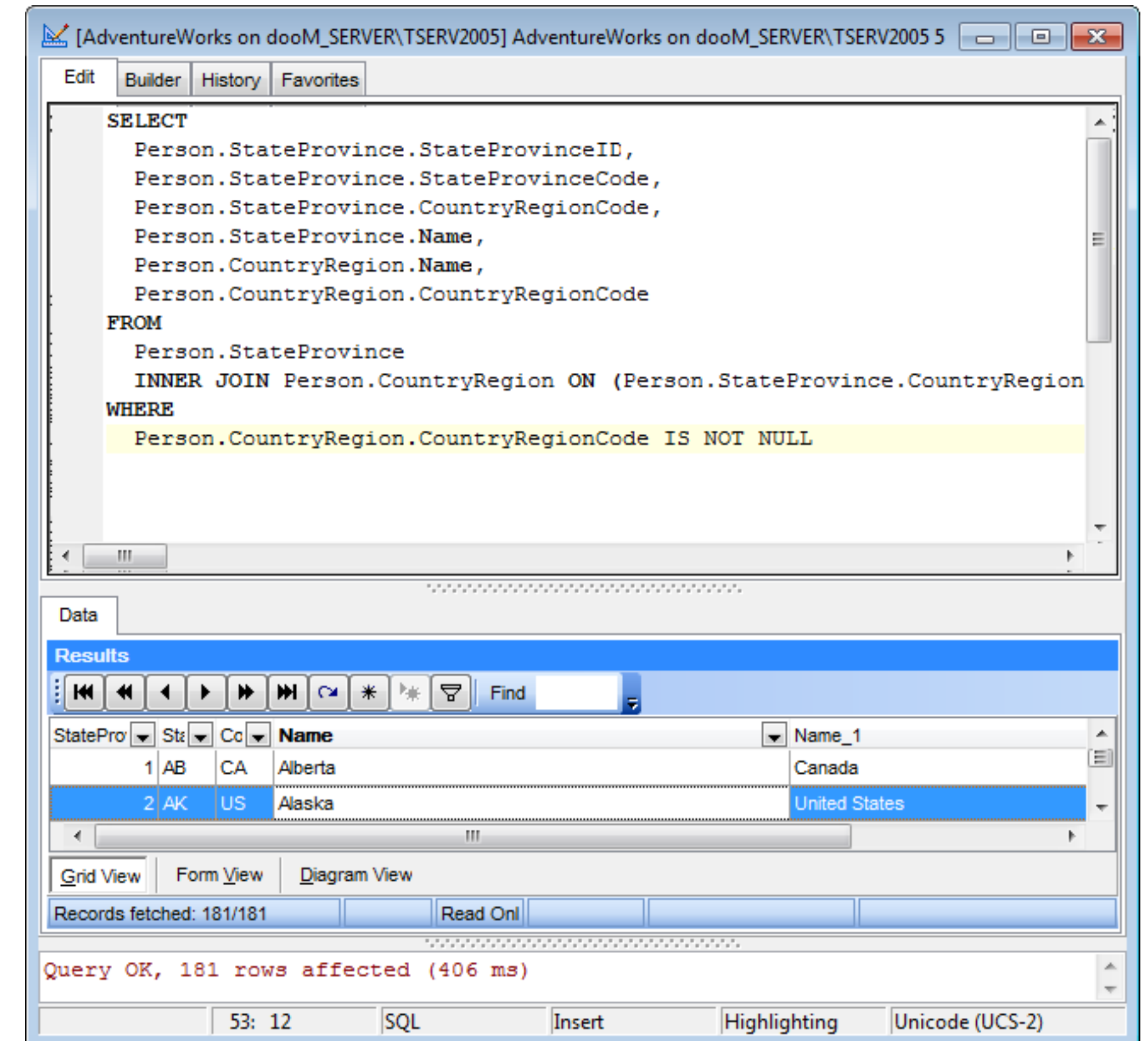
Id	Name	Surname
1	George	Washington
2	Thomas	Jefferson
3	John	Adams



President	Sucessor
1	3
3	2

Structured Query Language

- *Declarative* language
- You describe what you want, and a query engine figures out efficiency.
- **Quick raise of hands:** who knows SQL?



Anatomy of an SQL query

SELECT

Declare which columns you want to have in the resulting table.

FROM

Declare the tables that these columns come from, and how they should be joined together.

WHERE

Declare criteria for filtering rows in the resulting table.

GROUP BY

(Potentially) Create columns by arranging data into groups and then using aggregate functions to perform calculations in each group

For each century except the 21st, what is the average age difference between successive presidents?

```
SELECT      p1.century,  
            AVG(  
                ABS(DATEDIFF(year, p1.dob, p2.dob))  
            ) AS avg_diff  
FROM        succession s  
            JOIN presidents p1 ON P = p1.id  
            JOIN presidents p2 ON S = p2.id  
WHERE       p1.century != '21st'  
  
GROUP BY    p1.century
```


SELECT

p1.century,
AVG(
ABS(DATEDIFF(year, p1.dob, p2.dob))
) AS avg_diff

FROM

succession s
JOIN presidents p1 ON P = p1.id
JOIN presidents p2 ON S = p2.id

WHERE

WHERE p1.century != '21st'

GROUP BY

p1.century

id	name	surname	dob	Century	P	S
1	George	Washingto	2/22/1732	18th	1	3
2	Thomas	Jefferson	4/13/1743	19th	3	2
3	John	Adams	10/30/1735	18th	2	4
...	...	...	...	...	...	...
46	Joe	Biden	11/20/1942	21st	46	47
47	Donald	Trump	06/14/1946	21st	47	NaN

p1.id	p1.name	p1.surname	p1.dob	p1.century	s.P	s.S	p2.name	p2.surname	p2.dob	p2.century
1	George	Washington	2/22/1732	18th	1	3	John	Adams	10/30/1735	19th
2	Thomas	Jefferson	4/13/1743	19th	3	4	James	Madison	3/16/1751	19th
3	John	Adams	10/30/1735	19th	2	3	Thomas	Jefferson	4/13/1743	19th
...	...	...	...	...	...	...	...	...	...	...
46	Joe	Biden	11/20/1942	21st	46	47	Donald	Trump	06/14/1946	21st
47	Donald	Trump	06/14/1946	21st	47	NaN	NaN	NaN	NaN	NaN

SELECT

FROM

WHERE

GROUP BY

p1.century,
AVG(
ABS(DATEDIFF(year, p1.dob, p2.dob))
) AS avg_diff

succession s
JOIN presidents p1 ON P = p1.id
JOIN presidents p2 ON S = p2.id

WHERE p1.century != '21st'

p1.century

id	name	surname	dob	Century	P	S
1	George	Washingto	2/22/1732	18th	1	3
2	Thomas	Jefferson	4/13/1743	19th	3	2
3	John	Adams	10/30/1735	18th	2	4
...	...	...	...	...	...	...
46	Joe	Biden	11/20/1942	21st	46	47
47	Donald	Trump	06/14/1946	21st	47	NaN

p1.id	p1.name	p1.surname	p1.dob	p1.century	s.P	s.S	p2.name	p2.surname	p2.dob	p2.century
1	George	Washington	2/22/1732	18th	1	3	John	Adams	10/30/1735	19th
2	Thomas	Jefferson	4/13/1743	19th	3	4	James	Madison	3/16/1751	19th
3	John	Adams	10/30/1735	18th	2	3	Thomas	Jefferson	4/13/1743	18th
...	...	...	...	...	...	...	...	...	...	...
46	Joe	Biden	11/20/1942	21st	46	47	Donald	Trump	06/14/1946	21st
47	Donald	Trump	06/14/1946	21st	47	NaN	NaN	NaN	NaN	NaN

SELECT

FROM

WHERE

GROUP BY

p1.century,
AVG(
ABS(DATEDIFF(year, p1.dob, p2.dob))
) AS avg_diff

succession s
JOIN presidents p1 ON P = p1.id
JOIN presidents p2 ON S = p2.id

WHERE p1.century != '21st'

p1.century

id	name	surname	dob	Century	P	S
1	George	Washingto	2/22/1732	18th	1	3
2	Thomas	Jefferson	4/13/1743	19th	3	2
3	John	Adams	10/30/1735	18th	2	4
...	...	...	...	...	...	...
46	Joe	Biden	11/20/1942	21st	46	47
47	Donald	Trump	06/14/1946	21st	47	NaN

p1.id	p1.name	p1.surname	p1.dob	p1.century	s.P	s.S	p2.name	p2.surname	p2.dob	p2.century
1	George	Washington	2/22/1732	18th	1	3	John	Adams	10/30/1735	18th
3	John	Adams	10/30/1735	18th	2	3	Thomas	Jefferson	4/13/1743	19th


```
SELECT p1.century,
AVG(
  ABS(DATEDIFF(year, p1.dob, p2.dob))
) AS avg_diff
FROM succession s
JOIN presidents p1 ON P = p1.id
JOIN presidents p2 ON S = p2.id
WHERE p1.century != '21st'
GROUP BY p1.century
```

id	century	avg_diff
1	18th	5.5
...	...	...

p1.id	p1.name	p1.surname	p1.dob	p1.century
1	George	Washington	2/22/1732	18th
3	John	Adams	10/30/1735	18th

s.P	s.S
1	3
2	3

p2.name	p2.surname	p2.dob	p2.century
John	Adams	10/30/1735	18th
Thomas	Jefferson	4/13/1743	19th

SQL powers the world

- SQL-related concepts are key for most modern software handling data
- You should know the basics of SQL!
- The best way to learn is by doing exercises



```
df = succession.merge(
    presidents,
    left_on="president",
    right_on="id",
    how="inner"
).merge(presidents,
    left_on="successor",
    right_on="id",
    how="inner")

df = df[df["century"] != "21st"]

df["age_diff"] = (
    df["dob_x"] - df["dob_y"]
).dt.days.abs() / 365.25
```




VS



A Relational Database
Management System

Persistent storage on disk

Uses SQL

Data integrity, security,
scalability

A Python Data Analysis
Library

In-memory (volatile)

Uses SQL-flavored Python

Complex transformation,
data visualization, ML

Document data model

- Treats the world as a hierarchy of entities
- New information can be added to documents without affecting the structure of others.
- Querying languages (e.g., XPath) involve traversing the document tree!

XML

```
<contact>
  <firstname>Bob</firstname>
  <lastname>Smith</lastname>
  <phone type="Cell">(123) 555-0178</phone>
  <phone type="Work">(890) 555-0133</phone>
  <address>
    <type>Home</type>
    <street1>123 Back St.</street1>
    <city>Boys</city>
    <state>AR</state>
    <zip>32225</zip>
    <country>US</country>
  </address>
</contact>
```

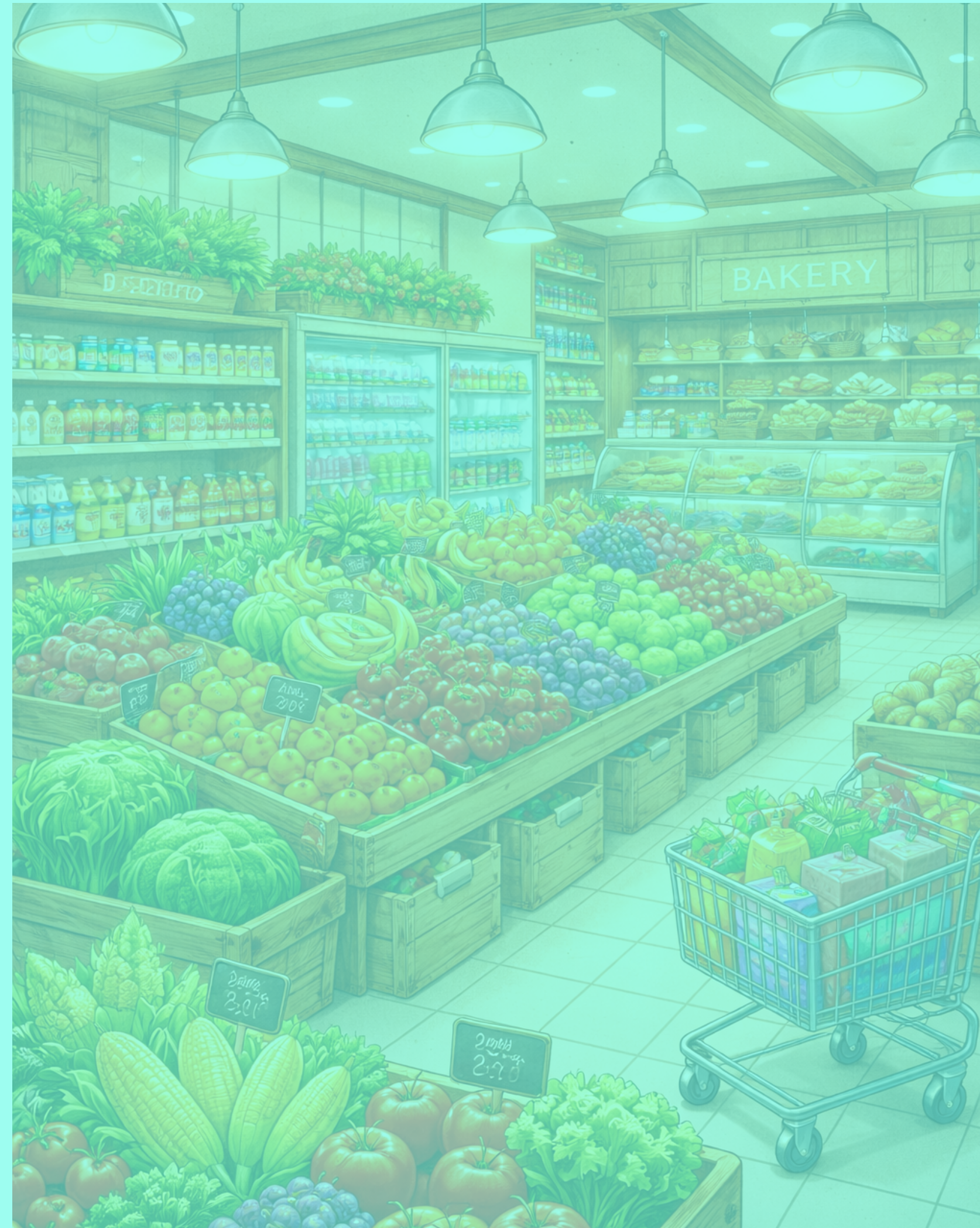
JSON

```
{
  "firstName": "Bob",
  "lastName": "Smith",
  "address": {
    "type": "Home",
    "street1": "5 Oak St.",
    "city": "Boys",
    "state": "AR",
    "zip": "32225",
    "country": "US"
  },
  "hobby": "sailing",
  "phone": {
    "type": "Cell",
    "number": "(555)-123-4567"
  }
}
```


Getting Data



Organizing Data



Wrangling Data



Data is a “moat”

- Companies (and researchers) gain a competitive advantage by having high-quality, hard-to-replicate data.
- But where does data come from? How to get it?

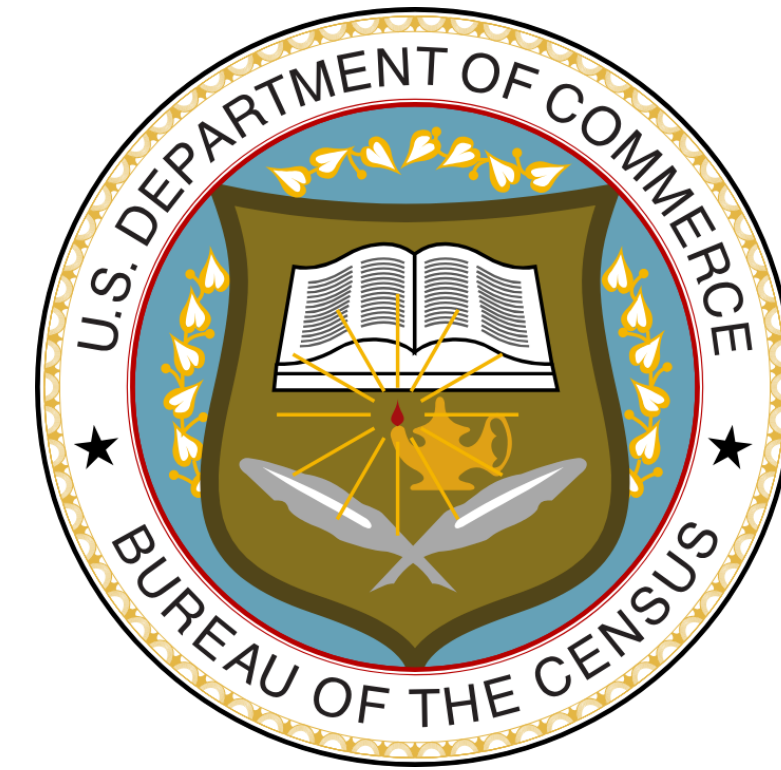


All sorts of institutions collect data

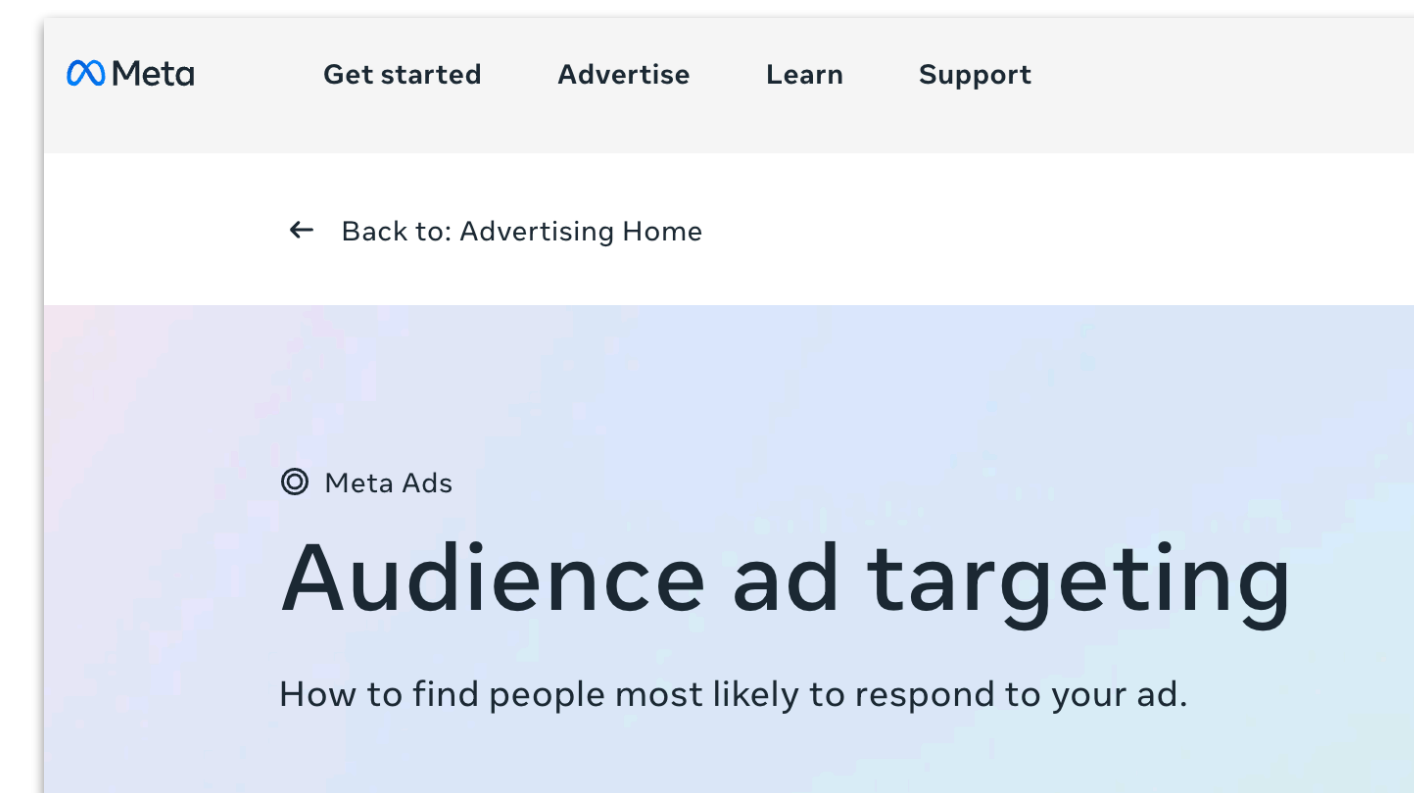
- Companies;
- Non-profits
- The government;



WIKIPEDIA
The Free Encyclopedia

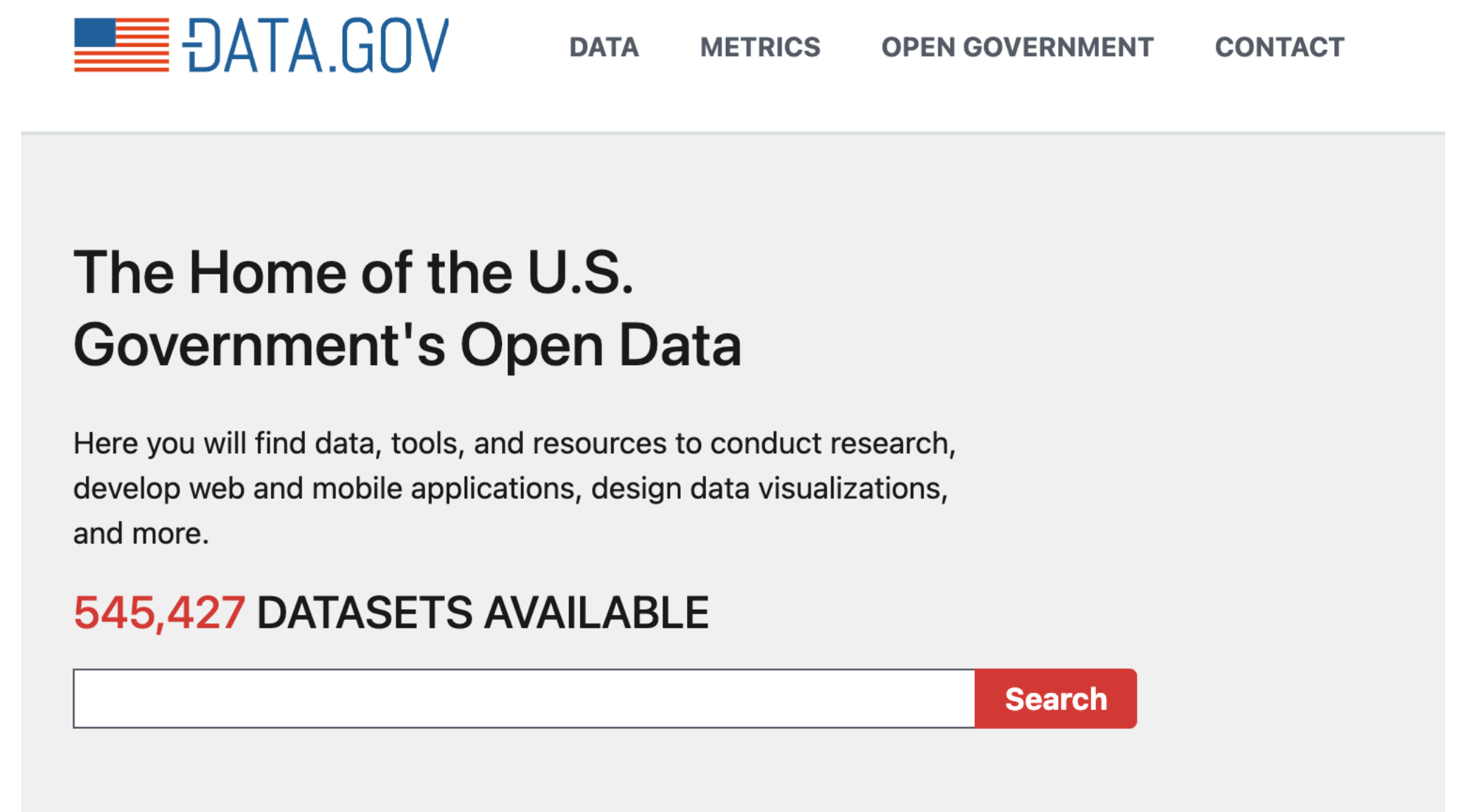


But... Sharing data is not
always in the best interest of
said institutions



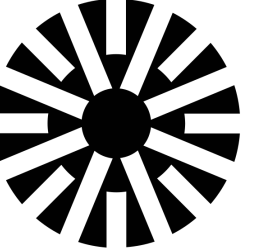
Governmental Data

- The government provides data for free, as a *public* good.
 - *E.g.*, the census, economic data, epidemiological data, and crime data.
- But also, *it provides private data* to certain institutions.
 - *E.g.*, tax history, detailed medical history, individual-level educational data.



Survey Research

- “Panels” of respondents that answer multiple “waves” of questions.
- You can buy their data (expensive) or ask them to collect specific data you care about (even more expensive).

Pew Research Center 

 Nielsen

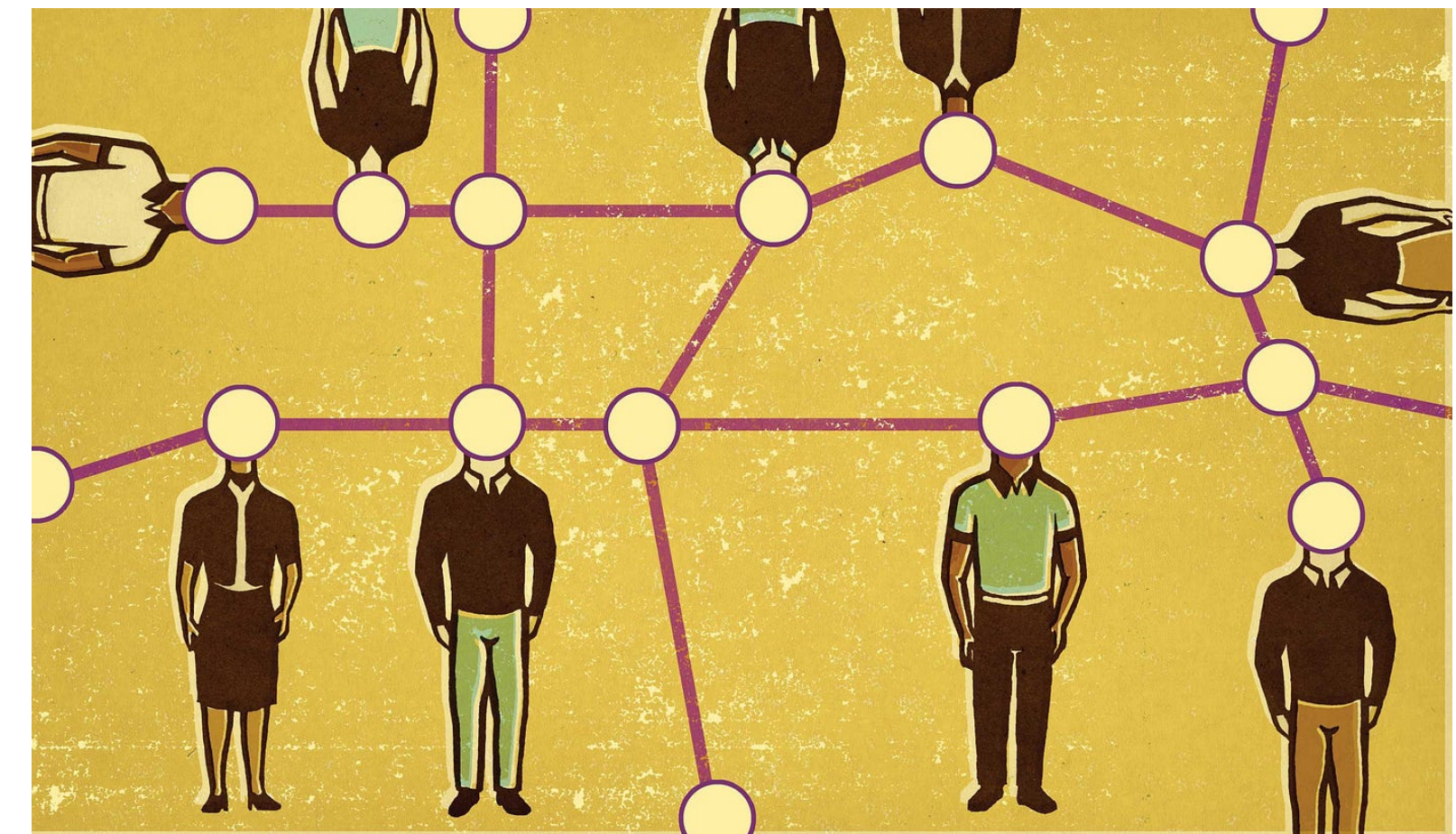
YouGov[®]

Crowdsourcing

- Platforms allow people to voluntarily participate in surveys and data annotation tasks.
- Many previous use cases can now be accomplished well with LLMs.
- Actual problem: people using LLMs to answer surveys!

 **CloudResearch**[®]

 **Prolific** **amazon**
mechanical turk



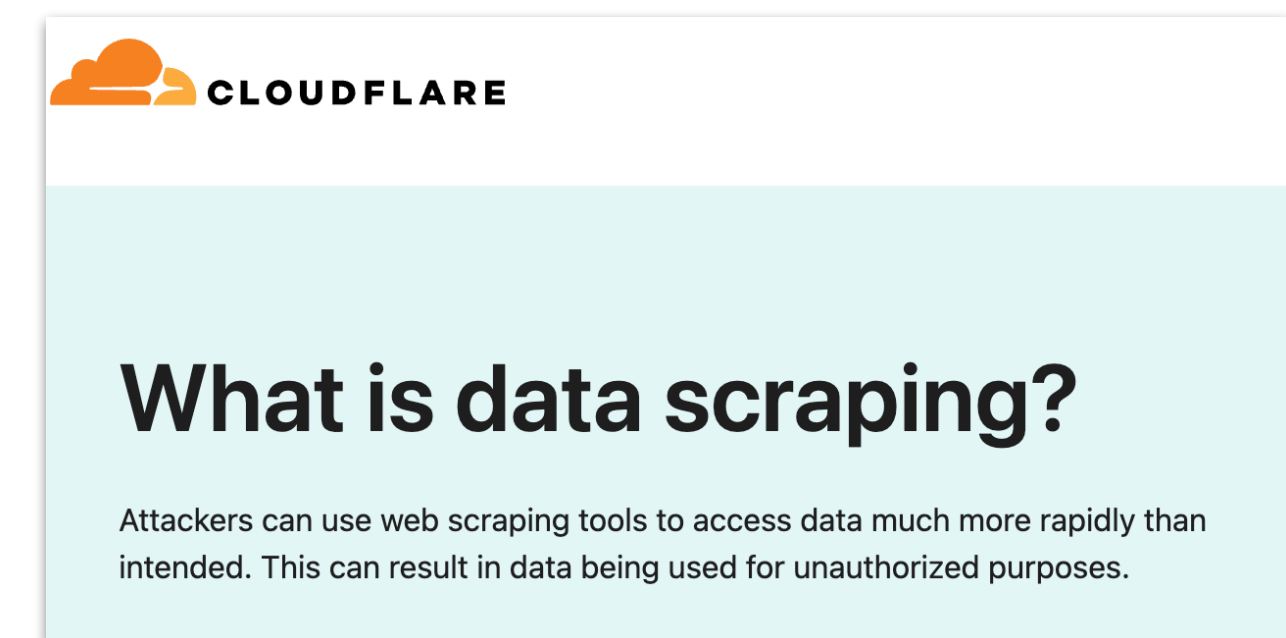
“Alt data market”

- Investment firms use various data to beat the market.
- Data brokers collect your data to sell to the government or to help target ads.



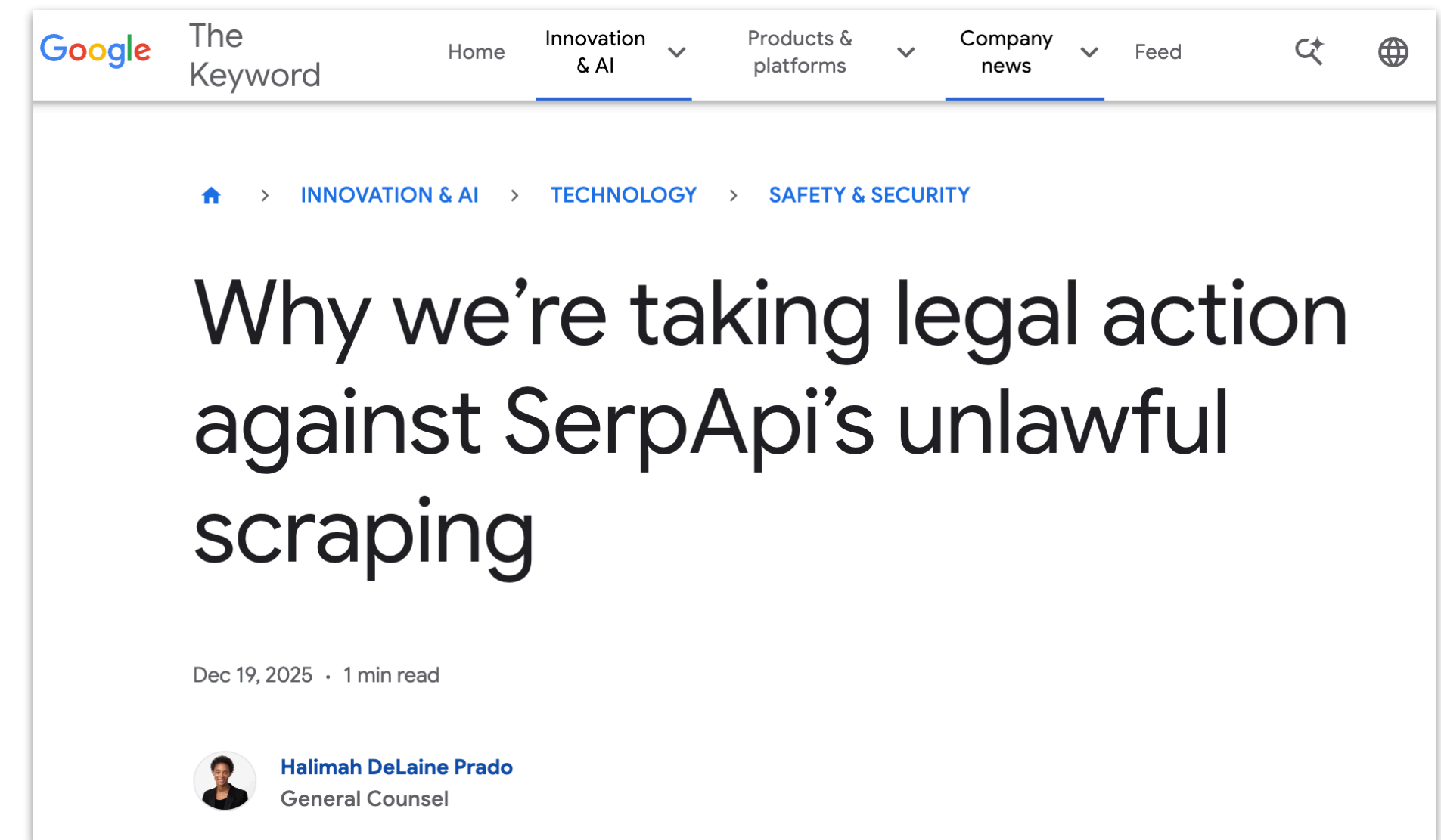
Scraping

- The Web has plenty of valuable information!
- But sometimes people don't want you to get it.
- The web is increasingly “closed,” i.e., non-public websites like social media.



Legal questions around scraping

- **Legal questions abound!**
 - Researchers do not necessarily have to follow ToS...
 - But these are murky waters.
 - Lots of recent cases!
 - *Google v. SerpApi*;
 - *Meta v. Bright Data Ltd.*
 - *OpenAI Copyright Infringement Litigation*)



(Link)

Other Researchers!

- **Beauty of research:**
academics get karma for publishing data!
- Open source repositories:
 - Hugging face
 - Zenodo
 - Harvard Dataverse
- Sometimes data agreement is needed for private data

zenodo



Hugging Face



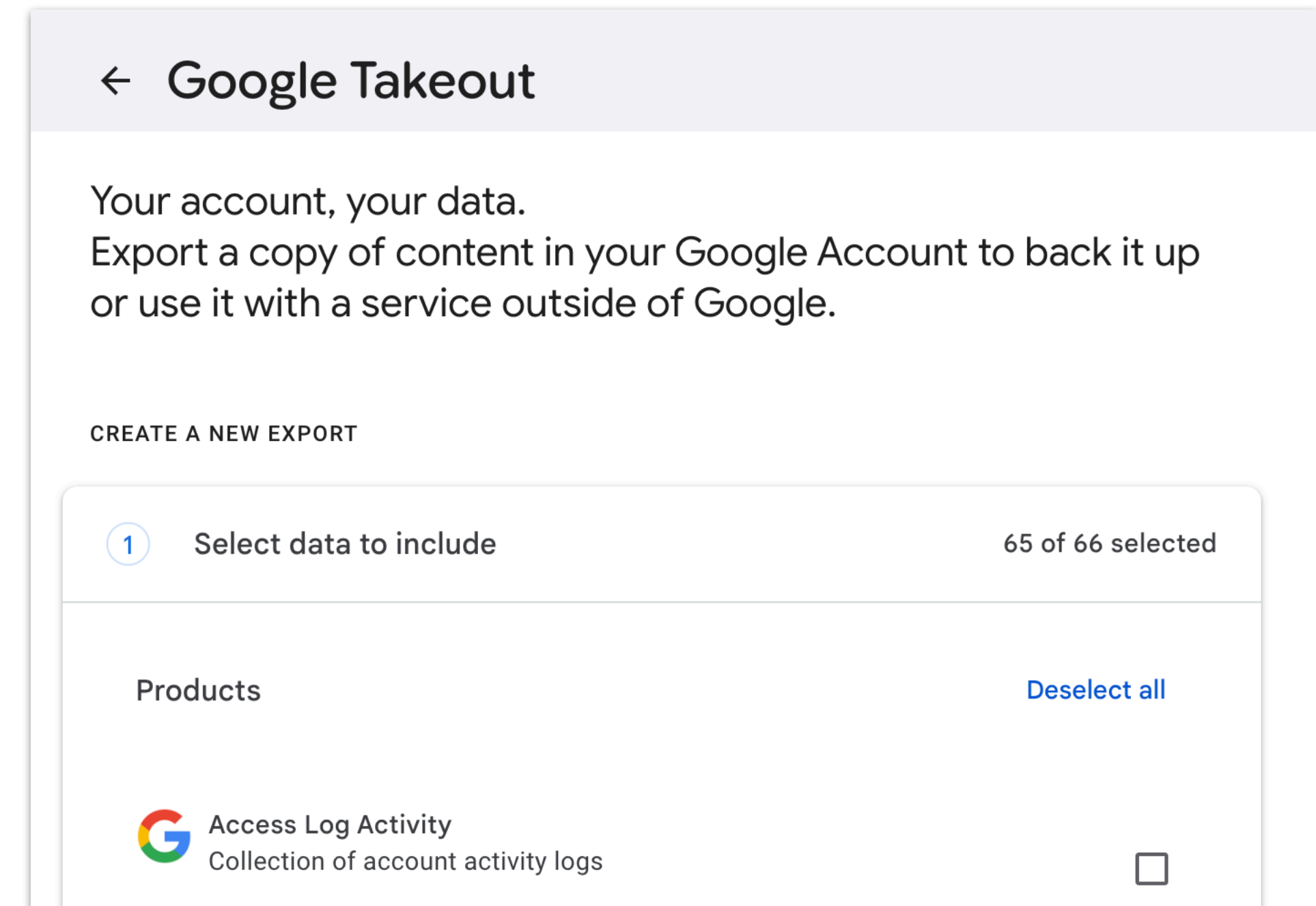
HARVARD
Dataverse

Other People!

Art. 20 GDPR – Right to data portability

- Platforms must allow EU residents to *export their data*.
- Companies usually extend this right to non-EU residents.

Example: Google Takeout



(Link)

Platforms may be forced to share data?

- The Digital Services Act in Europe
 - Large Online Platforms must share data with vetted researchers and authorities.
 - To reduce "systemic risks," e.g., disinformation and illegal content.
- Word in the street...
 - Super slow and bureaucratic...
 - Companies often say no...
 - Unclear how appeals work...



Getting Data



Organizing Data



Wrangling Data



Data Wrangling

- The process of turning data into a *useful form* for analysis, modeling, or visualization.
- Sounds like a trivial thing, but it actually comprises a sizable chunk of data work.

For Big-Data Scientists, 'Janitor Work' Is Key Hurdle to Insights

 Share full article  

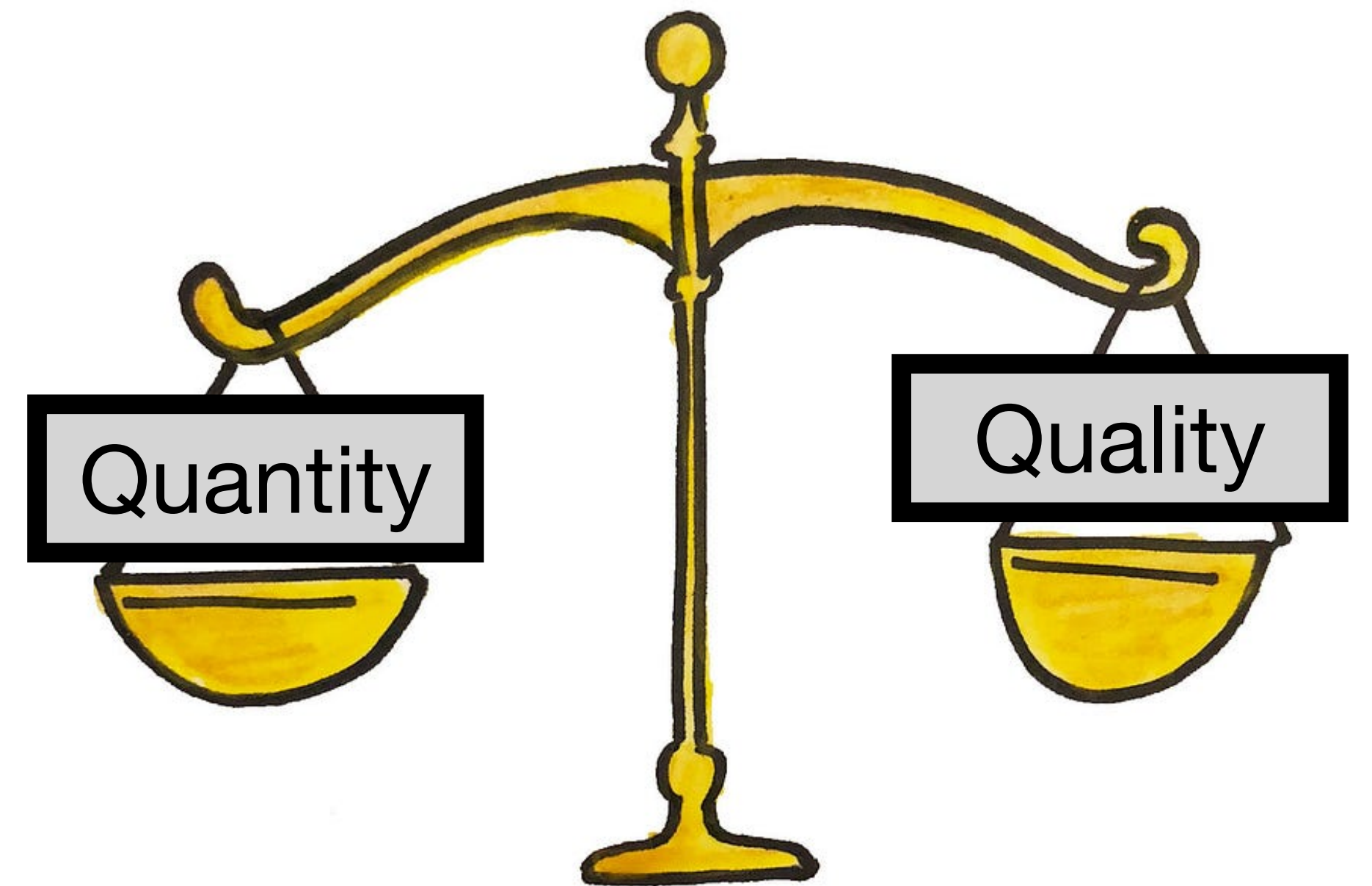


Monica Rogati, Jawbone's vice president for data science, with Brian Wilt, a senior data scientist. Peter DaSilva for The New York Times

[Souce]

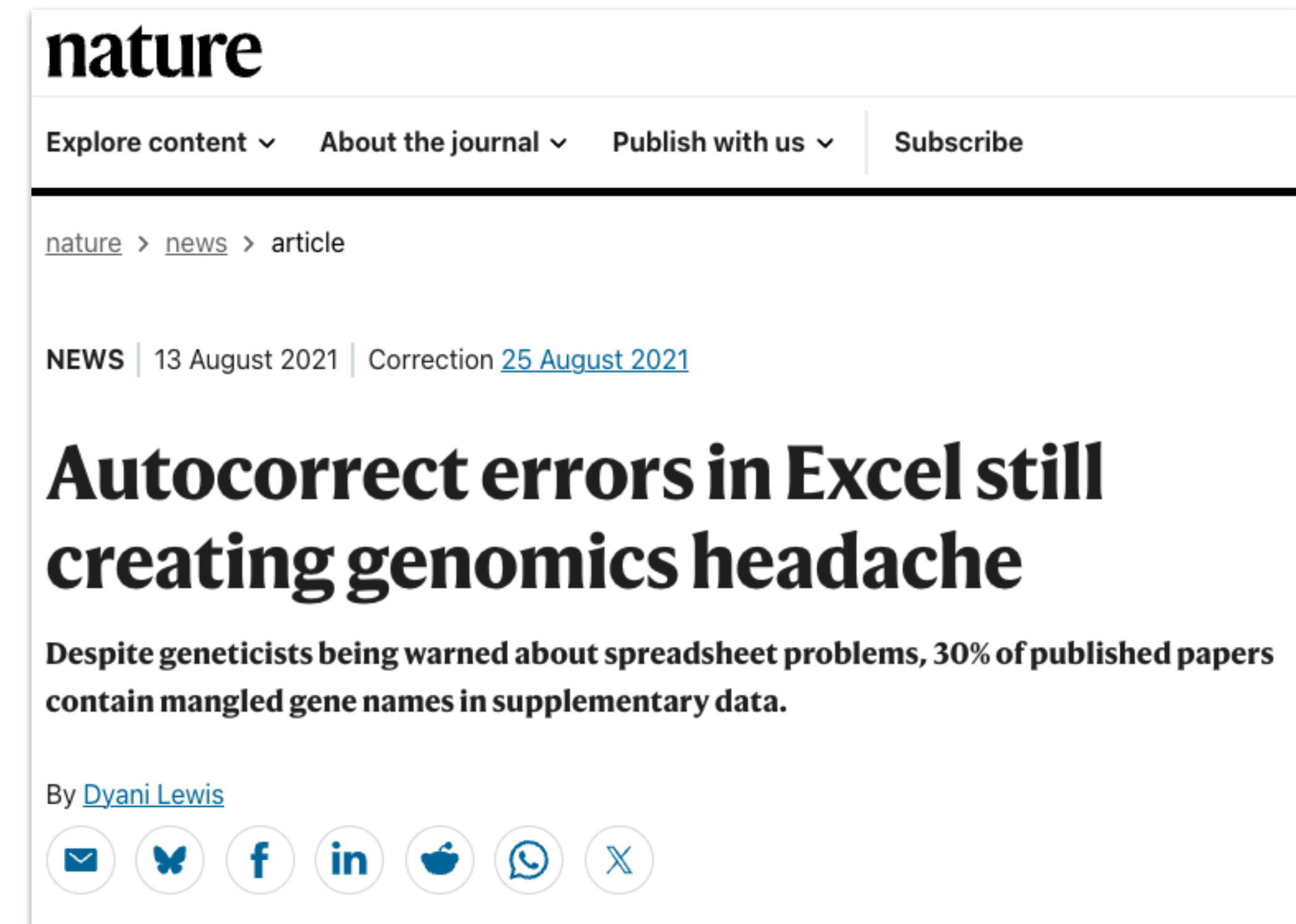
Working with Raw Data Sucks

- Data comes in all shapes and sizes
Common problems include:
 - Missing data (why?)
 - Incorrect data (how?)
 - Inconsistent data (where?)
- Oftentimes, cleaning data means you also lose valuable data.



Example from Bioinformatics

- Some genes have abbreviated forms similar to dates!
 - SEPT4 (Septin 4)
 - MARCH1 (membrane-associated ring-CH-type finger 1).
- When files are loaded in software like Excel, these genes are “renamed” as dates!

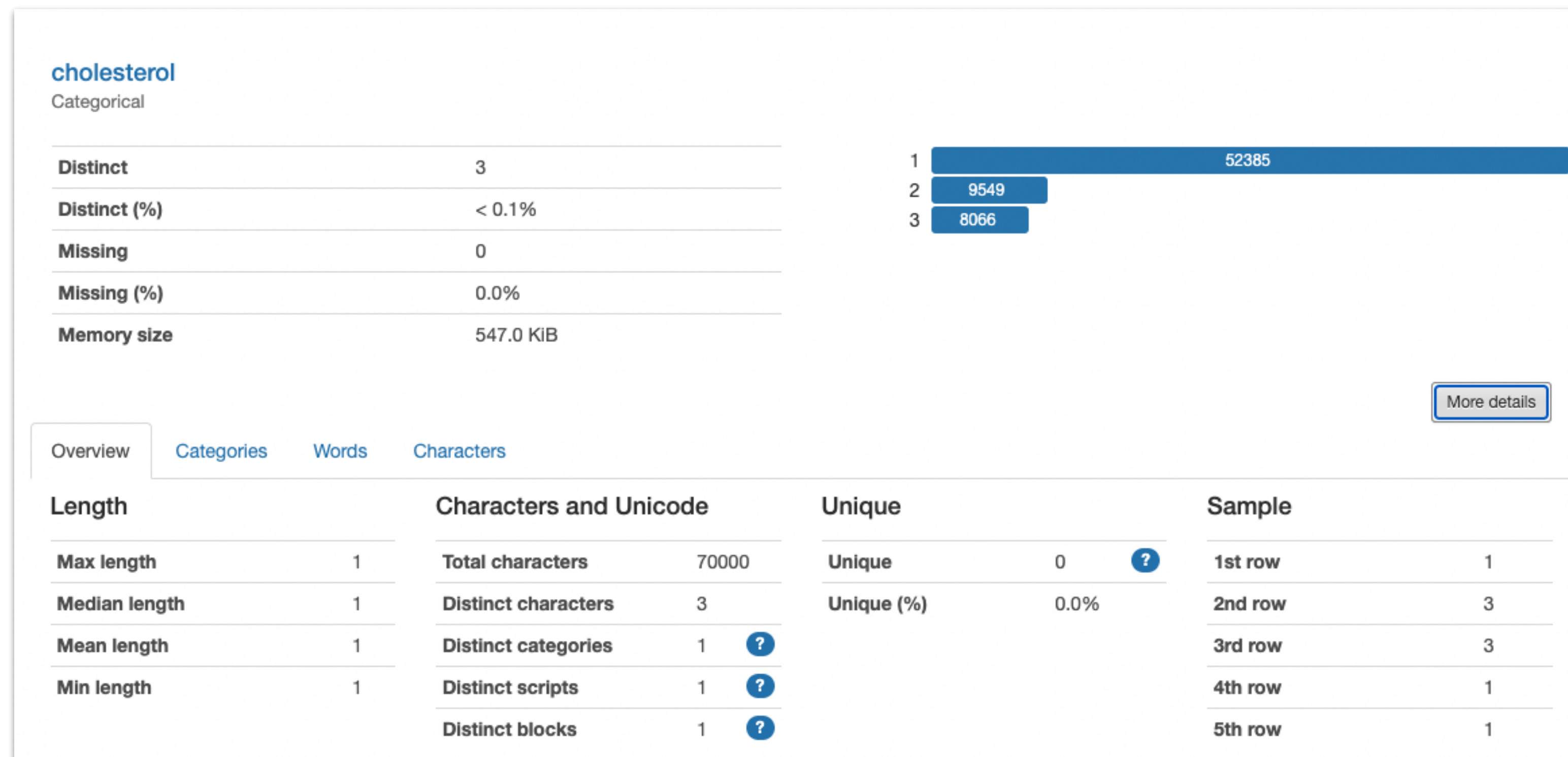


LLMs can be very helpful

- **Approach #1:** Write code to parse data!
- **Approach #2:** Use LLMs directly to parse messy text, HTML, etc.
- **Approach #3:** Standardize data.

```
.tk-jaf-domus { font-family: "jaf-domus", sans-serif; }
href=https://2019.jsconf.eu/ rel=canonical><link
href=/favicon-16x16.png rel=icon sizes=16x16 typ
name=theme-color><meta content=summary_large_ima
2019" name=twitter:title><meta content="JSConf E
JavaScript community in Europe." name=descriptio
Europe." name=twitter:description><meta content=
property=og:image><style>/ *! normalize.css v7.0.
decoration:none}.layout-wrapper,.section-video,.
normalize.css v7.0.0 | MIT License | github.com/
adjust:100%}article,footer,main,nav,section{dis
style:none;max-width:100%}svg:not(:root){overflo
appearance:button}[type=button]:-moz-focus-inne
moz-focusring,[type=reset]:-moz-focusring,[type=
box;box-sizing:border-box;padding:0}[type=number
offset:-2px}[type=search]:-webkit-search-cancel
appearance:button;font:inherit}.normal,h2,h3{fon
box-sizing:border-box;box-sizing:border-box}.br{
flexbox;display:flex}.flex-auto{-webkit-box-flex
items:center}.sans-serif{font-family:-apple-syst
family:Consolas,monaco,monospace}.times{font-fam
style:italic}.h2{height:32px;height:2rem}.h3{hei
in;transition:color .15s ease-in}.link:focus{out
type:none}.w2{width:32px;width:2rem}.ml3{margin-
align:center}.ttu{text-transform:uppercase}.nowr
variant:diagonal-fractions;line-height:1.666;let
title,h3{font-weight:700}.heading--massive{font-
spacing:normal}.h2,h2{line-height:1.625}.heading
```


Visualizations and basic stats can help diagnose data problems!



ydata-profiling

How to Ride No Handed on a Bike

- Wrangling data is a very practical skill.
- You will develop this during the project and the work.
- Hard for me to talk about it in a lecture.



Tidy Data

- Even processed datasets are often “messy.”
- Standardization proposal: tidy data!
 - Each variable has its own column
 - Each observation has its own row
 - Each type of observation has its table

Name	TreatA	TreatB
John	—	2
Jane	16	11
Mary	3	1

Name	Treatment	Result
John	A	—
John	B	2
Jane	A	16
Jane	B	11
Mary	A	3
Mary	B	1

Messy Data

Column headers are values, not variable names

religion	<\$10k	\$10–20k	\$20–30k	\$30–40k	\$40–50k	\$50–75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

Table 4: The first ten rows of data on income and religion from the Pew Forum. Three columns, \$75–100k, \$100–150k and >150k, have been omitted.

Messy Data

Multiple variables are stored in one column

country	year	column	cases
AD	2000	m014	0
AD	2000	m1524	0
AD	2000	m2534	1
AD	2000	m3544	0
AD	2000	m4554	0
AD	2000	m5564	0
AD	2000	m65	0
AE	2000	m014	2
AE	2000	m1524	4
AE	2000	m2534	4
AE	2000	m3544	6
AE	2000	m4554	5
AE	2000	m5564	12
AE	2000	m65	10
AE	2000	f014	3

(a) Molten data

country	year	sex	age	cases
AD	2000	m	0–14	0
AD	2000	m	15–24	0
AD	2000	m	25–34	1
AD	2000	m	35–44	0
AD	2000	m	45–54	0
AD	2000	m	55–64	0
AD	2000	m	65+	0
AE	2000	m	0–14	2
AE	2000	m	15–24	4
AE	2000	m	25–34	4
AE	2000	m	35–44	6
AE	2000	m	45–54	5
AE	2000	m	55–64	12
AE	2000	m	65+	10
AE	2000	f	0-14	3

(b) Tidy data

Messy Data

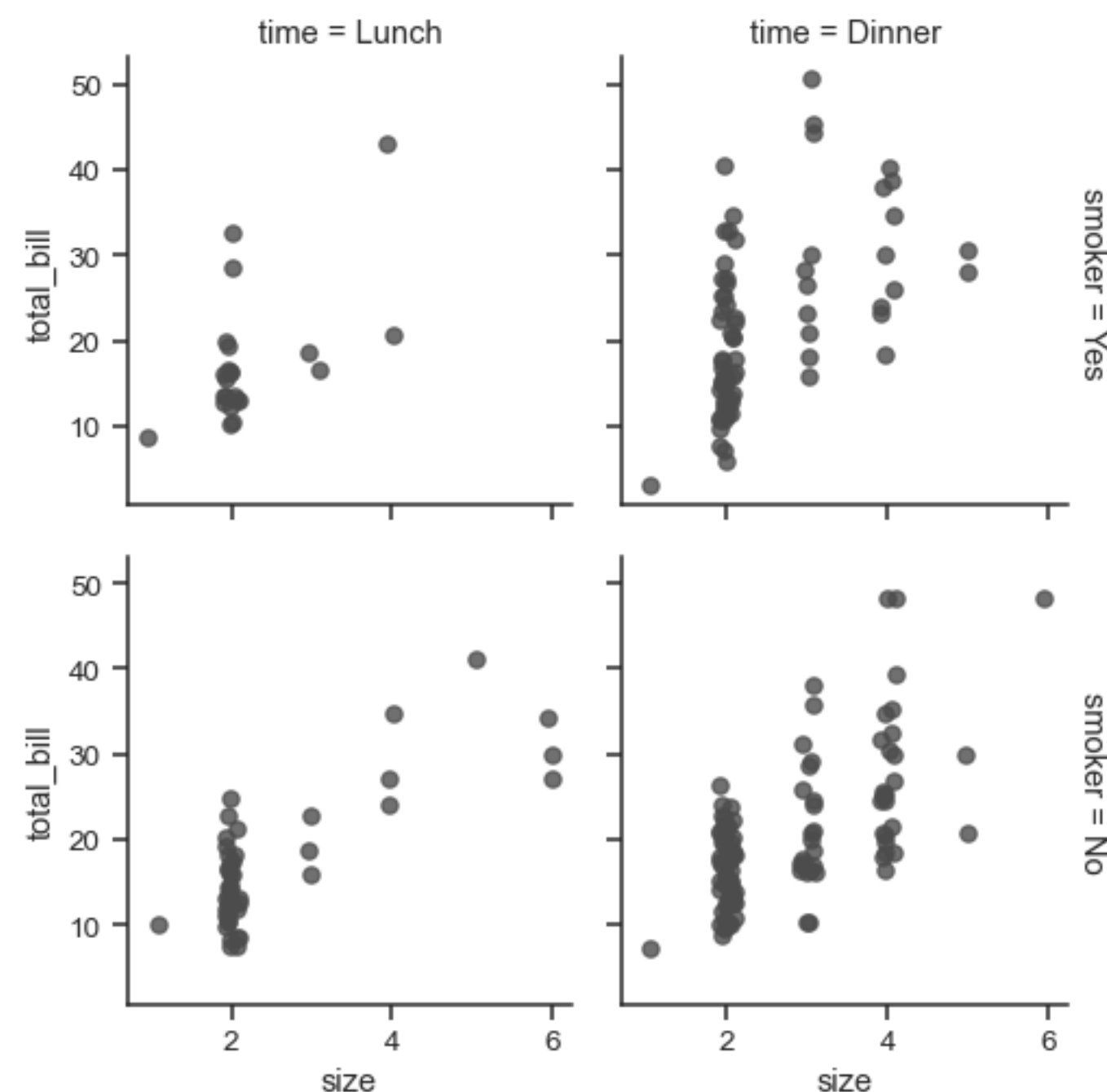
Variables are stored in both rows and columns

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	—	—	—	—	—	—	—	—

Table 11: Original weather dataset. There is a column for each possible day in the month. Columns d9 to d31 have been omitted to conserve space.

Why is Tidy Data Helpful?

It makes filtering, grouping, and plotting much easier!



total_bill	tip	sex	smoker	day	time	size
16.99	1.01	Female	No	Sun	Dinner	2
10.34	1.66	Male	No	Sun	Dinner	3
21.01	3.50	Male	No	Sun	Dinner	3
23.68	3.31	Male	No	Sun	Dinner	2
24.59	3.61	Female	No	Sun	Dinner	4
25.29	4.71	Male	No	Sun	Dinner	4

```
g = sns.FacetGrid(tips, row="smoker",  
col="time", margin_titles=True)  
  
g.map(sns.regplot, "size", "total_bill",  
color=".3", fit_reg=False, x_jitter=.1)
```


Two very useful operations

Melt

df3

	first	last	height	weight
0	John	Doe	5.5	130
1	Mary	Bo	6.0	150



df3.melt(id_vars=['first', 'last'])

	first	last	variable	value
0	John	Doe	height	5.5
1	Mary	Bo	height	6.0
2	John	Doe	weight	130
3	Mary	Bo	weight	150

Pivot

df

	foo	bar	baz	zoo
0	one	A	1	x
1	one	B	2	y
2	one	C	3	z
3	two	A	4	q
4	two	B	5	w
5	two	C	6	t



df.pivot(index='foo',
columns='bar',
values='baz')

bar	A	B	C
foo			
one	1	2	3
two	4	5	6

(Riding With a Dog is Also Hard)

Give us some
feedback!



[https://tinyurl.com/
reasoningwithdata](https://tinyurl.com/reasoningwithdata)