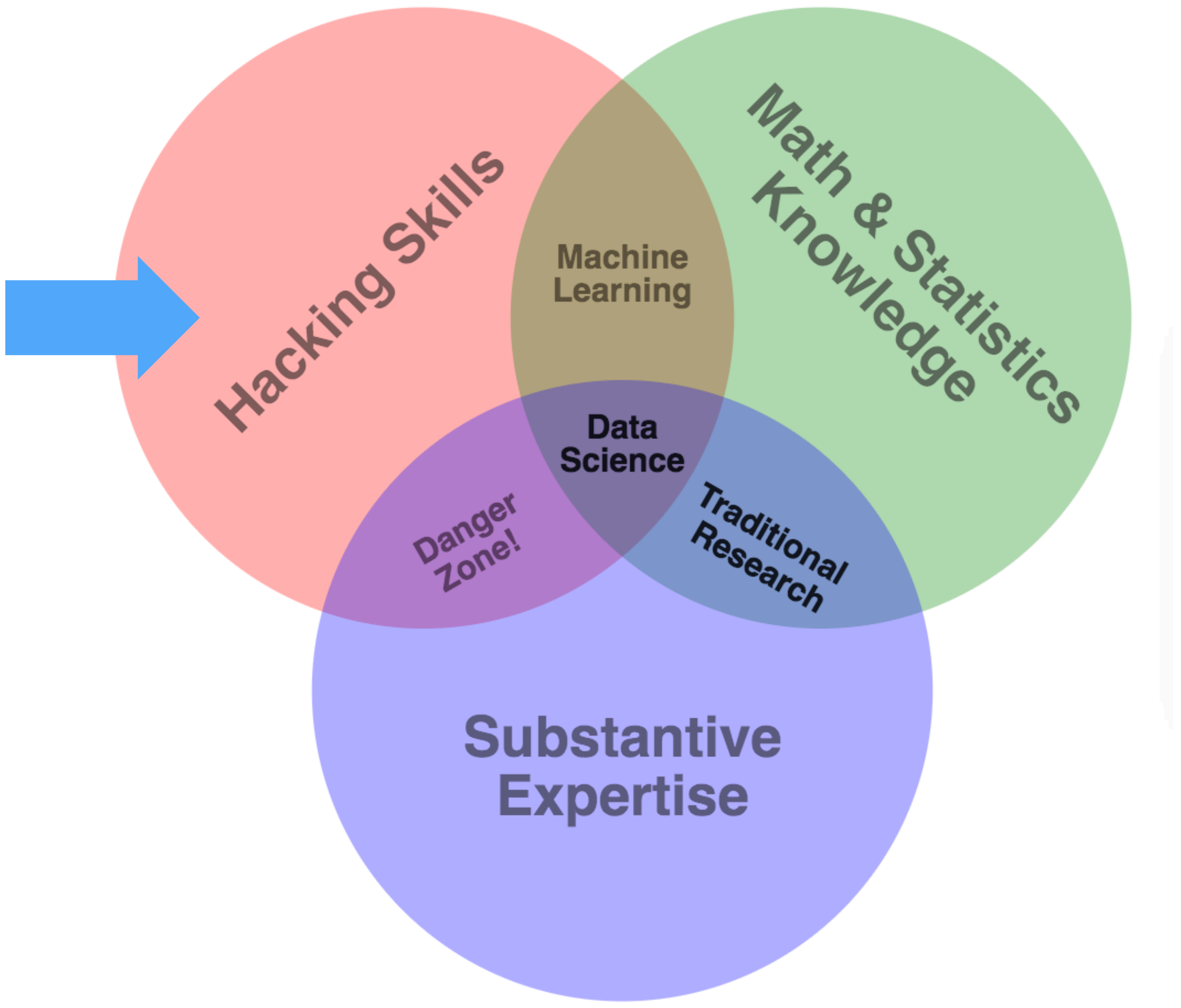
Lecture 3: Data Acquisition, Cleaning, Exploration

COSC 480 Data Science, Spring 2017
Michael Hay

we are (mostly) here



Credit: https://s3.amazonaws.com/aws.drewconway.com/viz/venn_diagram/data_science.html

Data acquisition

- Sources for data…
  - Existing databases, datasets
  - Logs: system logs, sensors, etc.
  - Web APIs
  - HTML scraping (if no API available and not forbidden by robots.txt)

The textbook provides a pretty good overview of scraping and using APIs. We will explore web scraping (a bit) in first lab.

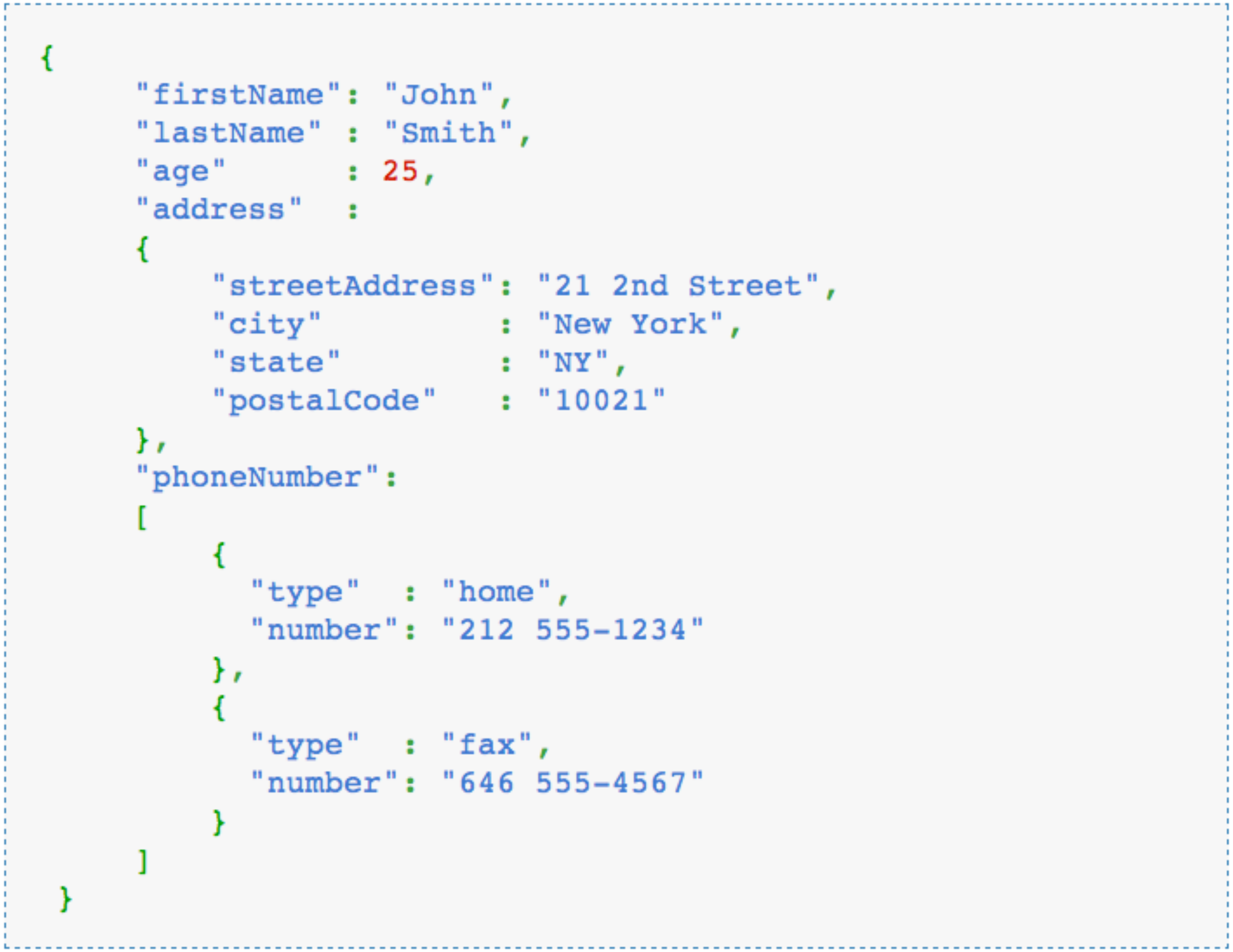
Data Formats

CSV

NYC Taxi Pickups and Dropoffs

```
VendorID,lpep_pickup_datetime,Lpep_dropoff_datetime,Store_and_fwd_flag,R
ateCodeID,Pickup_longitude,Pickup_latitude,Dropoff_longitude,Dropoff_lat
itude,Passenger_count,Trip_distance,Fare_amount,Extra,MTA_tax,Tip_amount
,Tolls_amount,Ehail_fee,improvement_surcharge,Total_amount,Payment_type,
Trip_type
2,2016-01-01 00:29:24,2016-01-01 00:39:36,N,
1,-73.928642272949219,40.680610656738281,-73.924278259277344,40.69804382
3242188,1,1.46,8,0.5,0.5,1.86,0,,0.3,11.16,1,1
2,2016-01-01 00:19:39,2016-01-01 00:39:18,N,
1,-73.952674865722656,40.723175048828125,-73.923919677734375,40.76137924
1943359,1,3.56,15.5,0.5,0.5,0,0,,0.3,16.8,2,1
2,2016-01-01 00:19:33,2016-01-01 00:39:48,N,
1,-73.971611022949219,40.676105499267578,-74.013160705566406,40.64607238
7695313,1,3.79,16.5,0.5,0.5,4.45,0,,0.3,22.25,1,1
2,2016-01-01 00:22:12,2016-01-01 00:38:32,N,
1,-73.989501953125,40.669578552246094,-74.000648498535156,40.68903350830
0781,1,3.01,13.5,0.5,0.5,0,0,,0.3,14.8,2,1
```

JSON (javascript object notation)



```
{
     "firstName": "John",
     "lastName" : "Smith",
     "age"      : 25,
     "address"  :
     {
         "streetAddress": "21 2nd Street",
         "city"         : "New York",
         "state"        : "NY",
         "postalCode"   : "10021"
     },
     "phoneNumber":
     [
         {
           "type"  : "home",
           "number": "212 555-1234"
         },
         {
           "type"  : "fax",
           "number": "646 555-4567"
         }
     ]
 }
```

XML (extensible markup language)

```
<data>
        <person id=“o555” >
                <name> Mary </name>
                <address>
                        <street> Maple </street>
                        <no> 345 </no>
                        <city> Seattle </city>
                </address>
        </person>
        <person>
                <name> John </name>
                <address> Thailand  </address>
                <phone> 23456  </phone>
        </person>
</data>
```

Relational Data

STUDENT

sid	name
1	Jill
2	Bo
3	Maya

Takes

sid	cid
1	445
1	483
3	435

COURSE

cid	title	sem
445	DB	F12
483	AI	S14
435	Arch	F12

PROFESSOR

fid	name
1	Diao
2	Saul
8	Weems

Teaches

fid	cid
1	445
2	483
8	435

Processing data

- CSV: pretty straightforward (use handy python tools, especially csv.DictReader)
- JSON, XML, HTML: tree-structured documents, requires a *parser (or a database, MongoDB).* Two flavors:
  - DOM: load in the whole tree (in memory!)
  - SAX: event-style (looks for open-close-tag events and calls back to user code to extract out interesting parts)
- Relational data: use a database! SQL! (Next week)

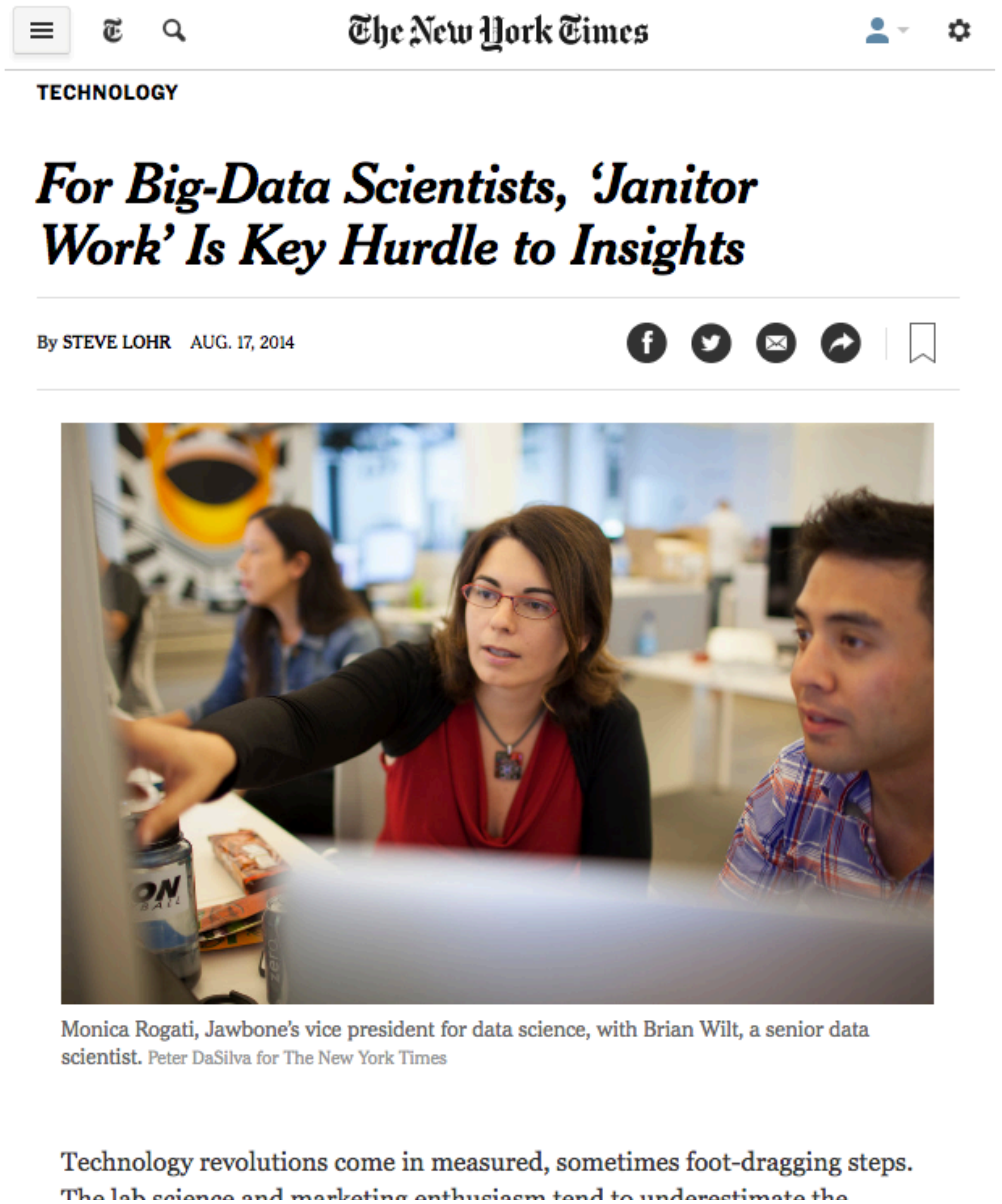
Data cleaning

Typical data quality issues

1. Parsing text into fields (separator issues)
2. Naming conventions: ER: NYC vs New York
3. Missing data
4. Different representations (2 vs Two)
5. Fields too long (get truncated)
6. Integrity violation (two people with the same social security number)
7. Redundant records (exact match or other)
8. Formatting issues – especially dates
9. Licensing issues/privacy/keep you from using the data as you would like

Credit: adapted from Berkeley CS 194 Lecture notes

Data cleaning: hard work!


The New York Times

TECHNOLOGY

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

By STEVE LOHR AUG. 17, 2014

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

Technology revolutions come in measured, sometimes foot-dragging steps. The lab science and marketing enthusiasm tend to underestimate the

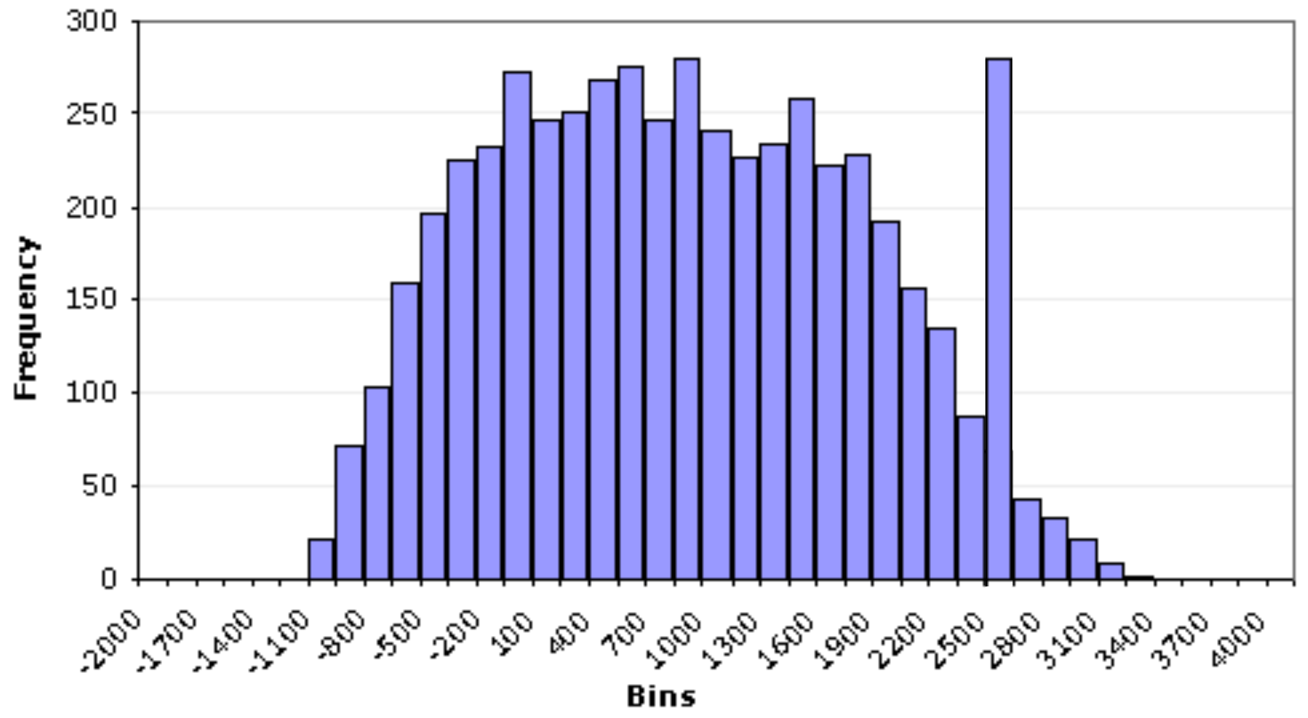
- *"Data scientists, according to interviews and expert estimates, spend from 50 percent to 80 percent of their time mired in this more mundane labor of collecting and preparing unruly digital data, before it can be explored for useful nuggets."*
- Several start ups…

https://www.nytimes.com/2014/08/18/technology/for-big-data-scientists-hurdle-to-insights-is-janitor-work.html

Lab 1!

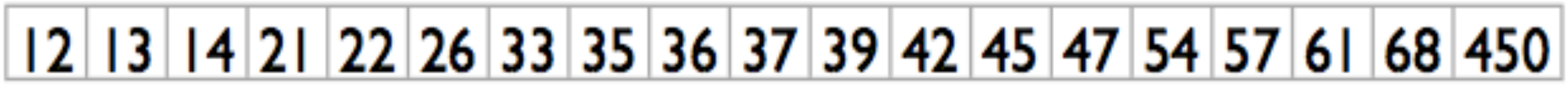
Data exploration

- Use ***visualization*** and ***basic statistics*** (min, max) to check for data quality issues:



Histograms show overall distribution, but small numbers of outliers may be hard to see

Domain knowledge is critical



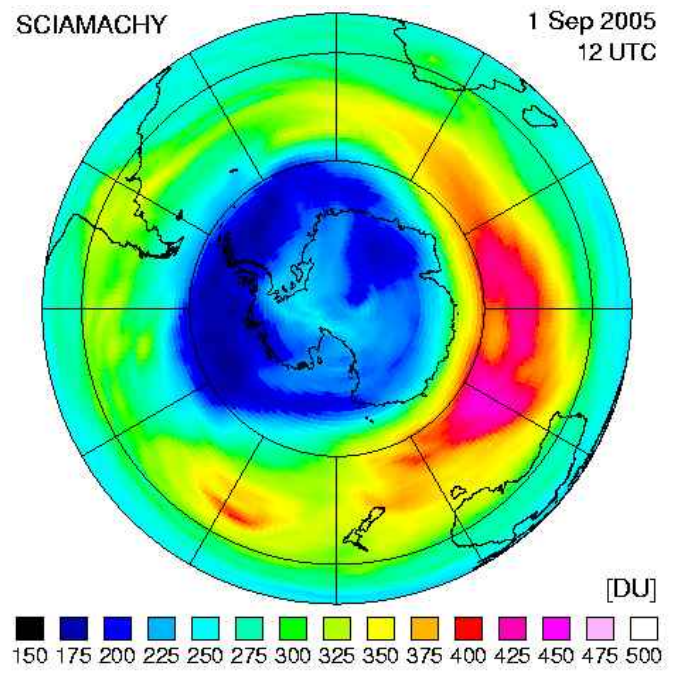
12	13	14	21	22	26	33	35	36	37	39	42	45	47	54	57	61	68	450

- Any anomalies in the data above?
  - If the data represents age of employees?
  - If the data represents no. of friends in social network?

Data cleaning gone too far?

"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***. When that explanation proved to be erroneous, they decided they were simply recording natural variations in the amount of ozone. It wasn't until 1985 that scientists were certain they were seeing a major problem."

https://www.ucar.edu/learn/1_6_1.htm



Regular Expressions

Regular Expressions

- A formal language to specify a set of strings
- Applications
  - Extract data fields from text
  - Search text/data for occurrences of patterns
- In Python, `import re`

Aside: raw string notation

- Backslash is a special character in Python

```
In [17]: s = 'Michael\'s string'

In [18]: print s
Michael's string
```

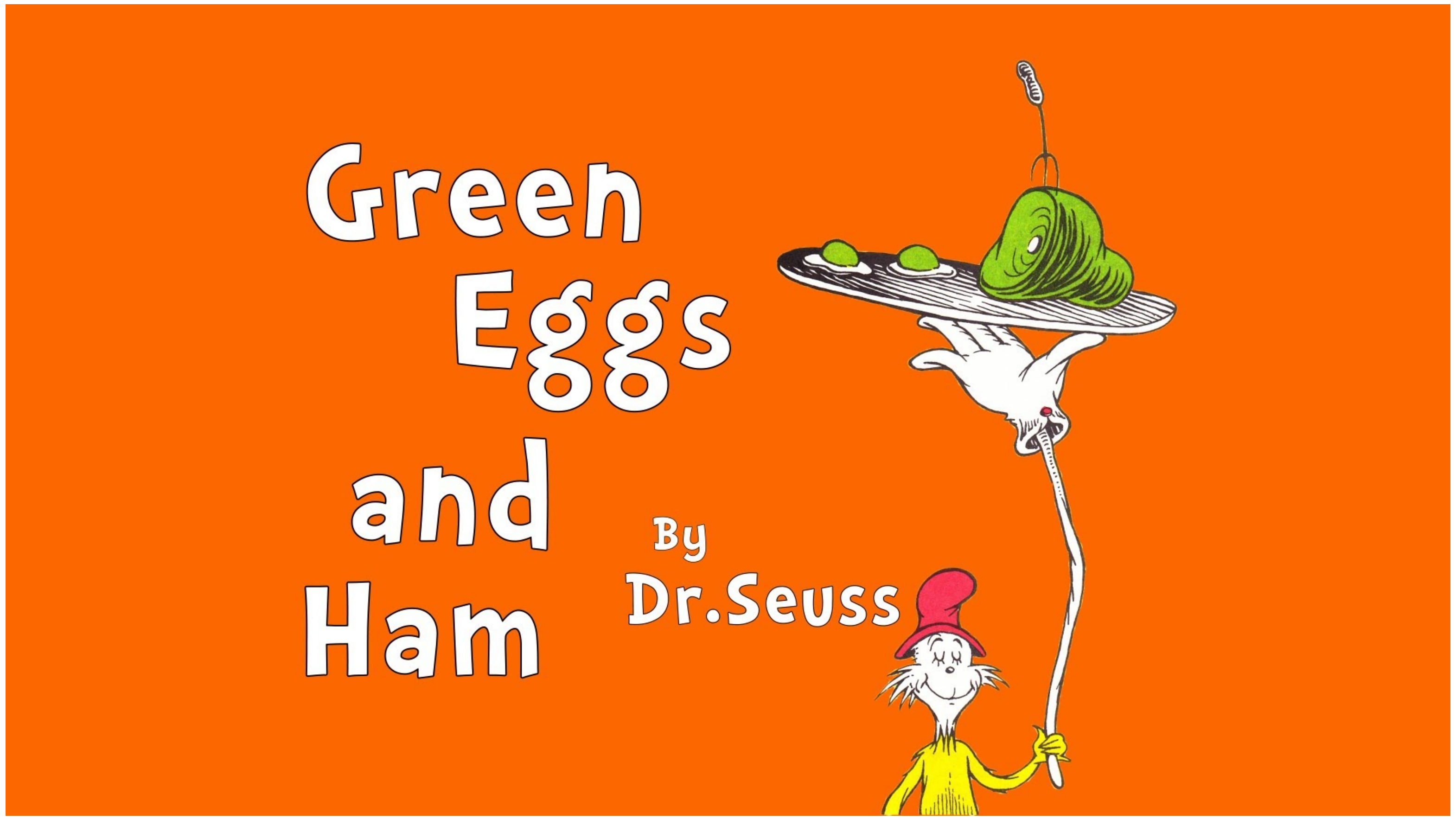
- But also a special character in regular expressions!

- Use *raw string notation* — `r'…'` — so Python interprets a backslash as a literal character.

```
In [19]: s = r'Michael\'s string'

In [20]: print s
Michael\'s string
```

Running example



Using https://regex101.com/

[] character sets

- Brackets define a set of characters. Can match any character in the set.
- Find all occurrences of "Sam" or "ham"?
- `r"[Sh]am"`

[character sets]

Expression	Meaning
r"[Sh]am"	Sam or ham
r"[A-Z]"	All upper case letters
r"[0-9]"	All numbers
r"[A-Za-z3-5]"	All letters as well as numbers between 3 and 5 inclusive
r"[\t\n]"	All white spaces

- How do find all occurrences of "Sam", "ham", or "am"? (Mind the dash!)
- `r"[Sh -]am"`
- To include "-" in your character set, put it last.

Disjunctions: |

- Find all occurrences of phone numbers (e.g., 315-228-7650)?
- What if you want to account for different variations in format?
  - "315-228-7650" or "(315) 228-7560" or …
- The "or" symbol | can be place between any two regular expressions.
- Find all occurrences of "eggs" or "ham"
- `r"eggs|ham"`

Negations [^]

- Find occurrences of rain but not train?

Would you, could you, in the rain?

I would not, could not, in the rain.
Not in the dark. Not on a train,

- `r"[^t]rain"`

Negations

Expression	Meaning
r"[^ \t\n]"	Non space characters
r"[^A-Z]"	Not capital letters
r"[^A-Za-z]"	Not letters

- *How do you search for '^' ?*

r"[\^]"

Regular operators: * + ? .

Expression	Meaning
r"no?body"	nobody , nbody
r"no*body"	nbody, nobody, noobody, nooobody, ….
r"no+body"	nobody, noobody, nooobody …
r"n.body"	n body, nobody, n3body, npbody …

- ? : 0 or 1 occurrence
- * : 0 or more occurrences
- + : 1 or more occurrences
- . : any one character

Start and End

- ^ : start of a string
- $: end of a string
- Note: what if string contains multiple lines of text and you want to capture start/end of *line*?
  - Turn on multiline mode.
- Example: Find all lines starting with "Not" and ending in period or exclamation point.

Not in a box.
Not with a fox.
Not in a house.
Not with a mouse.

Groups ()

- Parentheses are ***overloaded***
- Use to define subexpression:
  - Find all lines that start with "Not in…" or "Not with"?
  - `r"^Not (in|with)"`
- Use to capture groups: Find all the things Sam doesn't like…
  - `re.findall(r"I do not like (.*)$", s, re.MULTILINE)`

Exercise

Instructions: ~1 minute to think/ answer on your own; then discuss with neighbors; then I will call on one of you

- Find all occurrences of the word "am" in the text below (with apologies to Dr. Seuss). Solution should be case *insensitive* and ***not*** include Sam and ham.

I am Sam.
Sam I am!

Am I sam?
You bet your ham I am!

- `r"[^A-Za-z]([Aa]m)"`

Summary

- Cleaning is an important step before making sense of data.

- Diverse set of techniques are usually employed to fix many types of errors.

- Regular expressions are a useful tool to parse the data into the right format.