

PRINCIPLES OF DATA SCIENCE

JOHN P DICKERSON

Lecture #1 – 08/29/2018

**CMSC641
Wednesdays
7:00pm – 9:30pm**



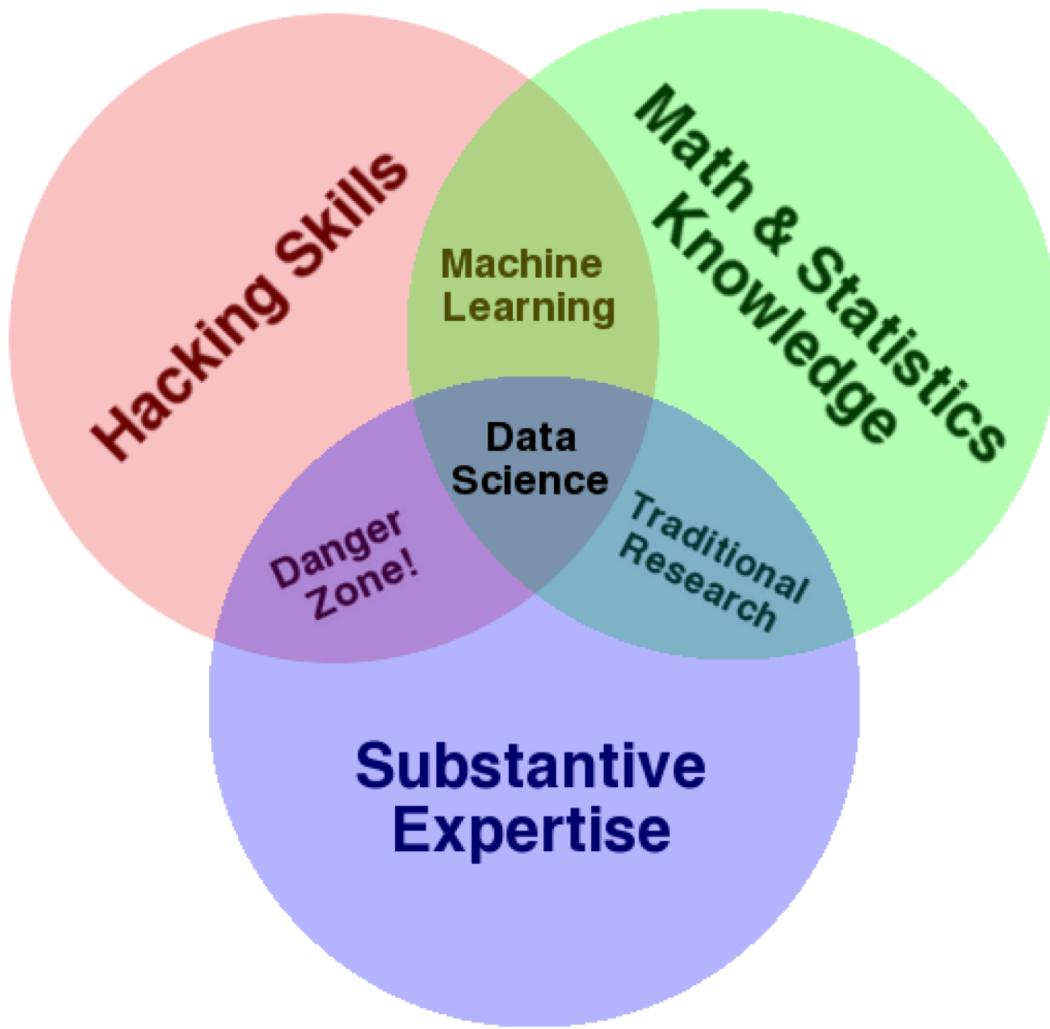
COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

INTRODUCTION TO

?????????????????

Data science is the application of **computational** and **statistical** techniques to address or gain [managerial or scientific] insight into some problem in the **real world**.

Zico Kolter
Machine Learning Prof, CMU



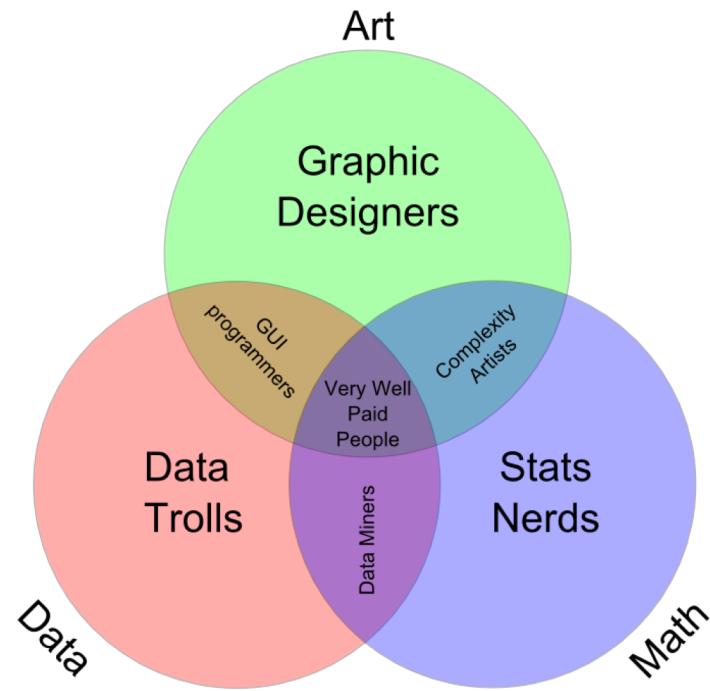
Drew Conway
CEO, Alluvium (analytics company)

MANY DEFINITIONS

Broad: necessarily **larger** than a single discipline

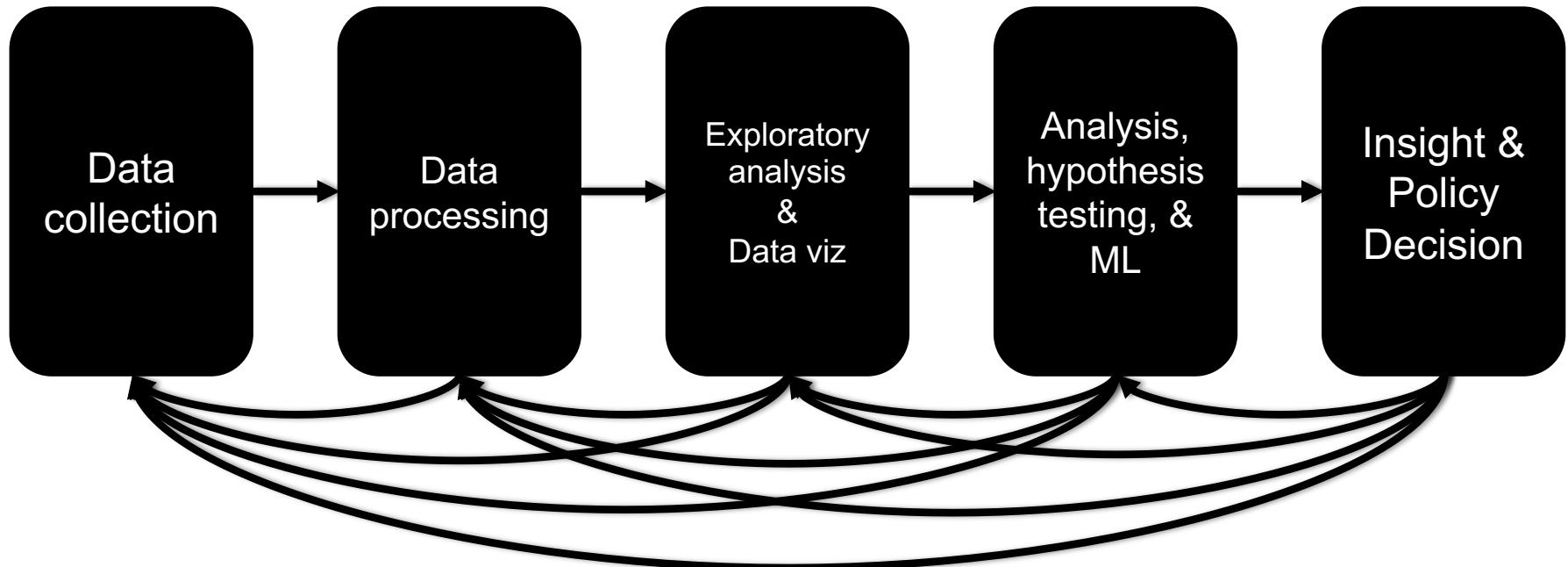
Interdisciplinary: statistics, computer science, operations research, statistical and machine learning, data warehousing, visualization, mathematics, information science, ...

Insight-focused: grounded in the desire to find insights in data and leverage them to inform decision making



Tuomas Carsey, UNC

THE DATA LIFECYCLE



“The ability to take data—to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it—that’s going to be a hugely important skill in the next decades, not only at the professional level but even at the educational level for elementary school kids, for high school kids, for college kids.”

Hal Varian
Chief Economist at Google

MOTIVATION

Explosion of data, in pretty much every domain

- Sensing devices and sensor networks that can monitor everything 24/7 from temperature to pollution to vital signs
- Increasingly sophisticated smart phones
- Internet, social networks makes it easy to publish data
- Scientific experiments and simulations → astronomical data volumes
- Internet of Things
- Dataification: taking all aspects of life and turning them into data (e.g., what you like/enjoy has been turned into a stream of your "likes")

How to handle that data? How to extract interesting actionable insights and scientific knowledge?

Data volumes expected to get much worse

FOUR V'S OF BIG DATA

Increasing data Volumes

- Scientific data: 1.5GB per genome -- can be sequenced in .5 hrs
- 500M tweets per day (as of 2013)
- As of 2012: 2.5 Exabytes of data created every day

Variety:

- Structured data, spreadsheets, photos, videos, natural text, ...

Velocity

- Sensors everywhere -- can generate high-rate "data streams"
- Real-time analytics requires data to be consumed as fast as it is generated

Veracity

- How do you decide what to trust? How to remove noise? How to fill in missing values?

THIS CERTIFICATE PROGRAM

You'll learn to take data:

- Process it
- Visualize it
- Understand it
- Communicate it
- Extract value from it



Hal Varian

Info: <https://www.cs.umd.edu/class/fall2018/cmsc641/>

Piazza: piazza.com/umd/fall2018/cmsc641

ELMS: (everyone should be registered automatically)

THIS CERTIFICATE PROGRAM

CMSC 641: Principles of Data Science

- An introduction to the data science pipeline, i.e., the end-to-end process of going from unstructured, messy data to knowledge and actionable insights. Provides a broad overview of what data science means and systems and tools commonly used for data science, and illustrates the principles of data science through several case studies.

CMSC 642: Big Data Systems

CMSC 643: Machine Learning and Data Mining

CMSC 644: Algorithms for Data Science

THIS CERTIFICATE PROGRAM

CMSC 641: Principles of Data Science

CMSC 642: Big Data Systems

- An overview of data management systems for performing data science on large volumes of data, including relational databases, and NoSQL systems. The topics covered include: different types of data management systems, their pros and cons, how and when to use those systems, and best practices for data modeling.

CMSC 643: Machine Learning and Data Mining

CMSC 644: Algorithms for Data Science

THIS CERTIFICATE PROGRAM

CMSC 641: Principles of Data Science

CMSC 642: Big Data Systems

CMSC 643: Machine Learning and Data Mining

- Provides a broad overview of key machine learning and data mining algorithms, and how to apply those to very large datasets. Topics covered include decision trees, linear models for classification and regression, support vector machines, neural networks and deep learning, online learning, recommendation systems, clustering and dimensionality reduction, and systems for large-scale machine learning.

CMSC 644: Algorithms for Data Science

THIS CERTIFICATE PROGRAM

CMSC 641: Principles of Data Science

CMSC 642: Big Data Systems

CMSC 643: Machine Learning and Data Mining

CMSC 644: Algorithms for Data Science

- Provides an in-depth understanding of some of the key data structures and algorithms essential for advanced data science. Topics include random sampling, graph algorithms, network science, data streams, and optimization.

THIS COURSE

End-to-end data science lifecycle

Acquiring, wrangling, cleaning, and integrating data; Setting up pipelines for ETL

Data modeling

Information Visualization

Ethics, Privacy, and Reproducibility

Feel free to tell me if there are topics that you think we should cover...

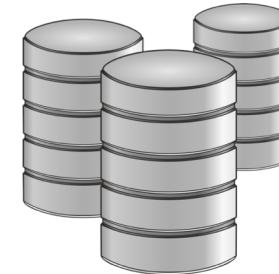
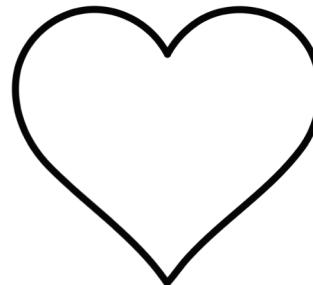
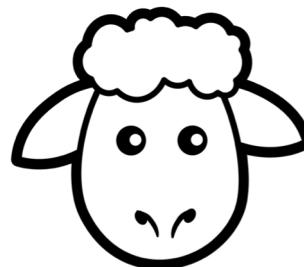
PREREQUISITE KNOWLEDGE

Aimed at **folks with some CS knowledge** – but likely accessible to others with programming experience and mathematical maturity.

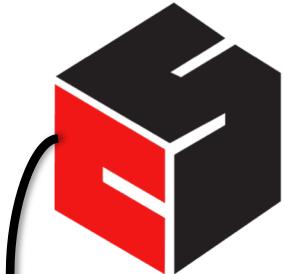
We **do not assume**:

- Experience with Python, pandas, scikit-learn, matplotlib, etc ...
- Deep statistics or any ML knowledge
- Database or distributed systems knowledge

We **do assume**: You want to be here!



WHO AM I?



COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Carnegie Mellon University
School of Computer Science



COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

<http://jpdkerson.com>



UNITED NETWORK FOR ORGAN SHARING



OPTIMIZED
MARKETS



**THE DIAN FOSSEY
GORILLA FUND®
INTERNATIONAL**

WHO ARE YOU?



STAT400?
CMSC422?
CMSC424?

Register on Piazza: piazza.com/umd/fall2018/cmsc641

(TENTATIVE) COURSE STRUCTURE

First 2 lectures: intro & primers in the Python data science stack

Next 3 lectures: data collection & management

- Best practices, data wrangling, exploratory analysis, ethics, debugging, visualization, etc ...

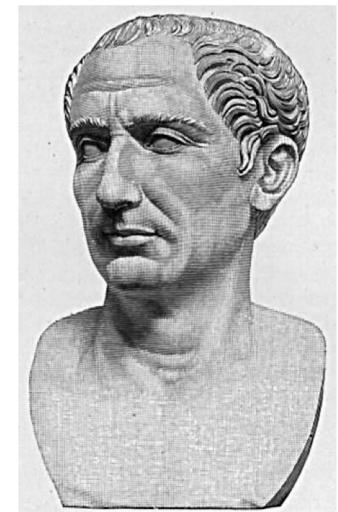
Next 4-5 lectures: statistical modeling & ML

- Statistical learning, regression, classification, cross-validation, model evaluation, hypothesis testing, etc ...

Midterm

Final 4.5 lectures: advanced topics

- Dimensionality reduction, distributed learning, big data, distributed computation
- *Either* group presentations or more lectures



Ambitious ...

GRADE #1: MINI-PROJECTS

Students will complete **four mini-project assignments:**

- **Case studies** meant to mimic what you, a future data scientist, will see in industry. They should be fun ☺.

The rules:

- Allowed: small group **discussions**
- Required: individual **programming & writing**
- Never allowed: public posting of solutions

Deliverable:

- Turn in an .ipynb of a Jupyter notebook on ELMS

GRADE #2: READING HOMEWORKS

We will post (bi)weekly reading assignments. Mix of:

- Blog posts
- Academic articles
- News articles

Weekly quiz to be taken on ELMS covering the readings

Individual quiz grades are pass/fail:

- At least 60% correct → Pass
- Less than 60% correct → Fail

Must take at least ten of these quizzes over the semester

GRADE #3: MINI-TUTORIAL

In lieu of a final exam, you'll create a mini-tutorial that:

- Identifies a raw data source
- Processes and stores that data
- Performs exploratory data analysis & visualization
- Derives insight(s) using statistics and ML
- Communicates those insights as actionable text

Individual or group project

Will be hosted publicly online (GitHub Pages) and will strengthen your portfolio.



READY-MADE DATASET REPOSITORIES

<https://www.data.gov/>

- US-centric agriculture, climate, education, energy, finance, health, manufacturing data, ...

<https://cloud.google.com/bigquery/public-data/>

- BigQuery (Google Cloud) public datasets (bikeshare, GitHub, Hacker News, Form 990 non-profits, NOAA, ...)

<https://www.kaggle.com/datasets>

- Microsoft-owned, various (Billboard Top 100 lyrics, credit card fraud, crime in Chicago, global terrorism, world happiness, ...)

<https://aws.amazon.com/public-datasets/>

- AWS-hosted, various (NASA, a bunch of genome stuff, Google Books n-grams, Multimedia Commons, ...)

NEW DATASET IDEAS



Fraternal Order of Police vs Black Lives Matter

Linking finance data to \${anything_else}

Something having to do with Pokémon statistics?

Look through <http://www.alexa.com/topsites> and scrape something interesting!

University of Maryland-related, or College Park-related, stuff

- Check out <http://umd.io/> – open source project; maybe your data collection and cleaning scripts can be added to this!

Honestly, pretty much anything! Just document everything.

Reproducibility!

FINAL TUTORIAL

Deliverable: URL of your own GitHub Pages site hosting an .ipynb/.html export of your final tutorial

- <https://pages.github.com/> – make a GitHub account, too!
- <https://github.com/blog/1995-github-jupyter-notebooks-3>

The project itself:

- ~1500+ words of Markdown prose
- ~150+ lines of Python
- Should be viewable as a **static webpage** – that is, if I (or anyone else) opens the link up, everything should render and I shouldn't have to run any cells to generate output

FINAL TUTORIAL RUBRIC

The TAs and I will grade on a scale of 1-10:

Motivation: Does the tutorial make the reader believe the topic is important (a) in general and (b) with respect to data science?

Understanding: After reading the tutorial, does the reader understand the topic?

Further resources: Does the tutorial “call out” to other resources that would help the reader understand basic concepts, deep dive, related work, etc?

Prose: Does the prose in the Markdown portion of the .ipynb add to the reader’s understanding of the tutorial?

Code: Does the code help solidify understanding, is it well documented, and does it include helpful examples?

Subjective Evaluation: If somebody linked to this tutorial from Hacker News, would people actually read the whole thing?

GRADE #😊: CLASS PARTICIPATION

**Please please please please please do the required reading,
if available, before coming to class!**

Earn full credit via:

- Lecture participation
- Piazza participation
- Regular attendance at office hours (can be just to chat!)

*Please I'm so lonely ...
And so are the TAs ...*

**Aim to ask/answer a question at least once every two weeks;
or attend office hours at least once a month**

GRADE BREAKDOWN

60% mini-projects:

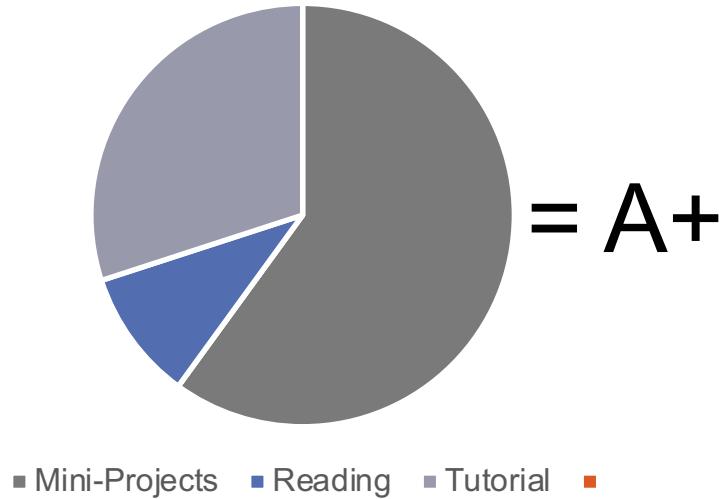
- There are 3 of them
- Equal weighting @ 20% each

10% reading homeworks

30% final tutorial

(+5% course participation)

Your Grade

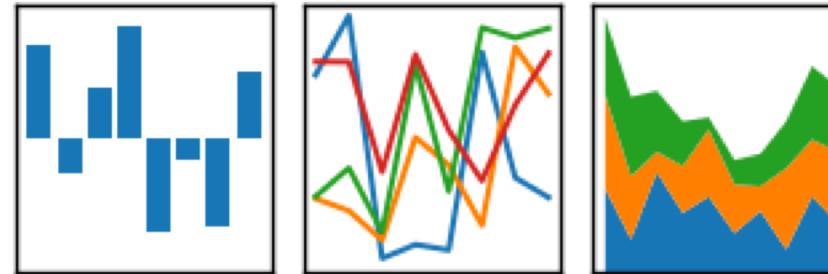


SOME TECHNOLOGIES WE WILL USE

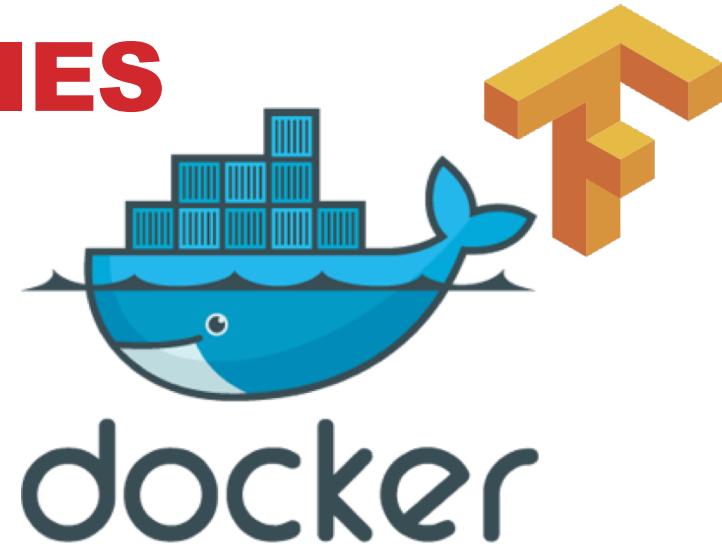


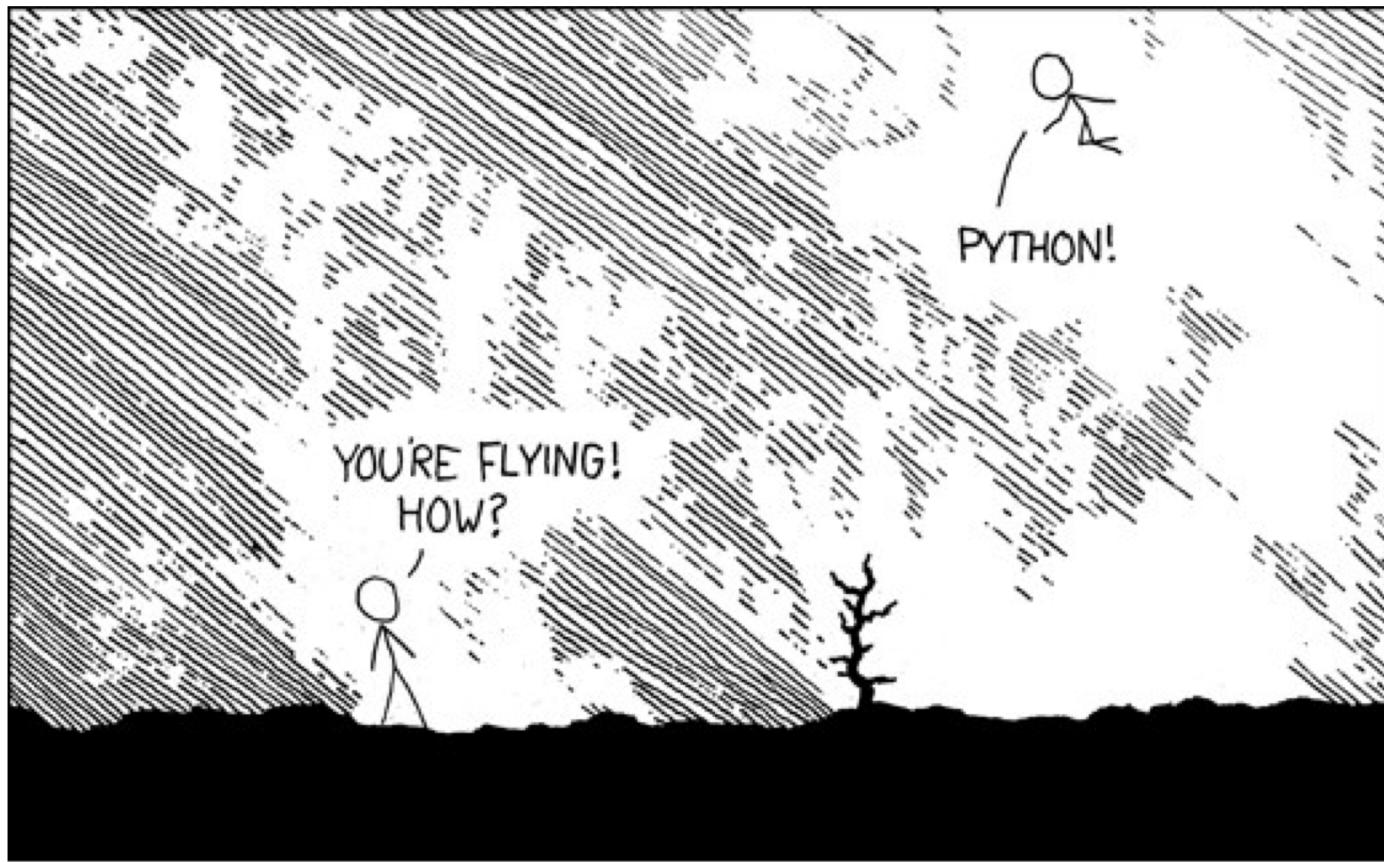
pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Spark





(Don't tell CMSC330 ...)

IMPORTANT WALLS OF TEXT

Common
Sense!

ANTI-HARASSMENT

(Adapted from ACM SIGCOMM's policies)

The open exchange of ideas and the freedom of thought and expression are central to our aims and goals. These require an environment that recognizes the inherent worth of every person and group, that fosters dignity, understanding, and mutual respect, and that embraces diversity. For these reasons, we are dedicated to providing a harassment-free experience for participants in (and out) of this class.

Harassment is unwelcome or hostile behavior, including speech that intimidates, creates discomfort, or interferes with a person's participation or opportunity for participation, in a conference, event or program.

Common
Sense!

ACADEMIC INTEGRITY

(Text unironically stolen from Hal Daumé III)

Any assignment or exam that is handed in must be your own work (unless otherwise stated). However, talking with one another to understand the material better is strongly encouraged. Recognizing the distinction between cheating and cooperation is very important. If you copy someone else's solution, you are cheating. If you let someone else copy your solution, you are cheating (this includes *posting solutions online in a public place*). If someone dictates a solution to you, you are cheating.

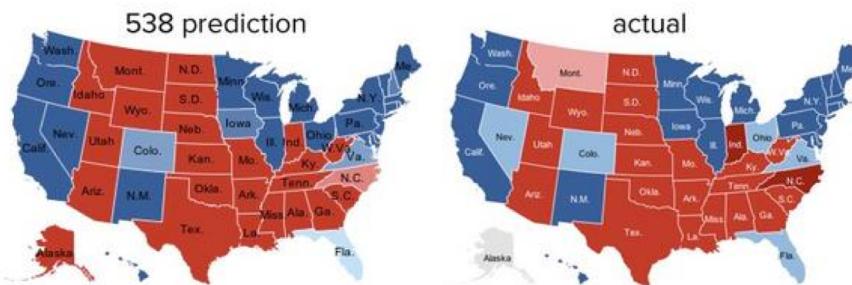
Everything you hand in must be in your own words, and based on your own understanding of the solution. If someone helps you understand the problem during a high-level discussion, you are not cheating. We strongly encourage students to help one another understand the material presented in class, in the book, and general issues relevant to the assignments. When taking an exam, you must work independently. Any collaboration during an exam will be considered cheating. Any student who is caught cheating will be given an F in the course and referred to the University Office of Student Conduct. Please don't take that chance – if you're having trouble understanding the material, please let me know and I will be more than happy to help.

(A FEW) DATA SCIENCE SUCCESS STORIES & CAUTIONARY TALES

POLLING: 2008 & 2012

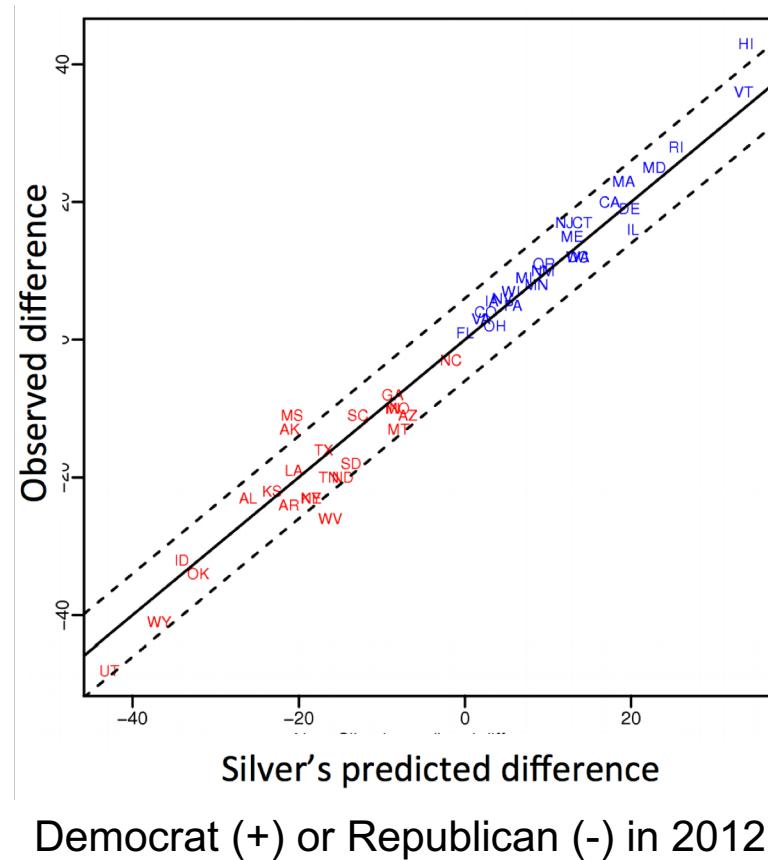
Nate Silver uses a simple idea – taking a principled approach to aggregating polling instead of relying on punditry – and:

- Predicts 49/50 states in 2008
- Predicts 50/50 states in 2012



- (He is also a great case study in creating a brand.)

<https://hbr.org/2012/11/how-nate-silver-won-the-2012-p>



Democrat (+) or Republican (-) in 2012

POLLING: 2016

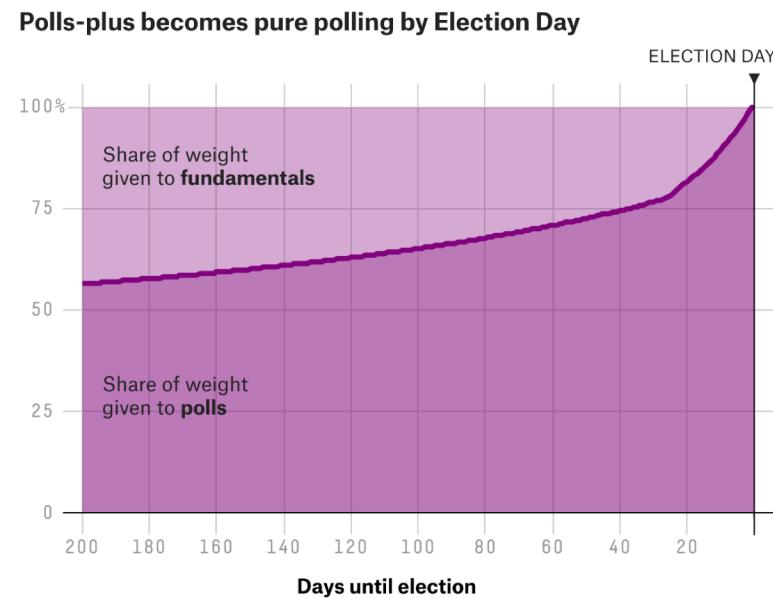
POLITICS

Nate Silver Is Unskewing Polls – All Of Them – In Trump's Direction

The vaunted 538 election forecaster is putting his thumb on the scales.

HuffPo: “He may end up being right, but he’s just guessing. A “trend line adjustment” is merely political punditry dressed up as sophisticated mathematical modeling.”

538: Offers quantitative reasoning for re-/under-weighting older polls, & changing as election approaches



http://www.huffingtonpost.com/entry/nate-silver-election-forecast_us_581e1c33e4b0d9ce6fbc6f7f
<https://fivethirtyeight.com/features/a-users-guide-to-fivethirtyeights-2016-general-election-forecast/>

AD TARGETING

Pregnancy is an **expensive & habit-forming time**

- Thus, valuable to consumer-facing firms

2012:

- Target identifies 25 products and subsets thereof that are commonly bought in early pregnancy
- Uses purchase history of patrons to predict pregnancy, targets advertising for post-natal products (cribs, etc)
- Good: increased revenue
- Bad: this can **expose** pregnancies – as famously happened in Minneapolis to a high schooler



AUTOMATED DECISIONS OF CONSEQUENCE

[Sweeney 2013, Miller 2015, Byrnes 2016, Rudin 2013, Barry-Jester et al. 2015]



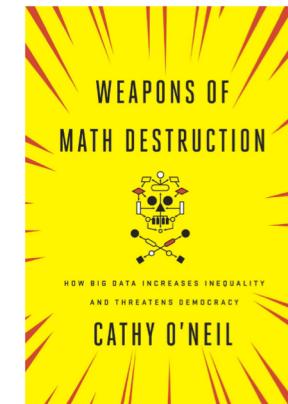
Hiring

Search for minority names →
ads for DUI/arrest records

Female cookies →
less freq. shown professional job opening ads

Lending

Policing/ sentencing

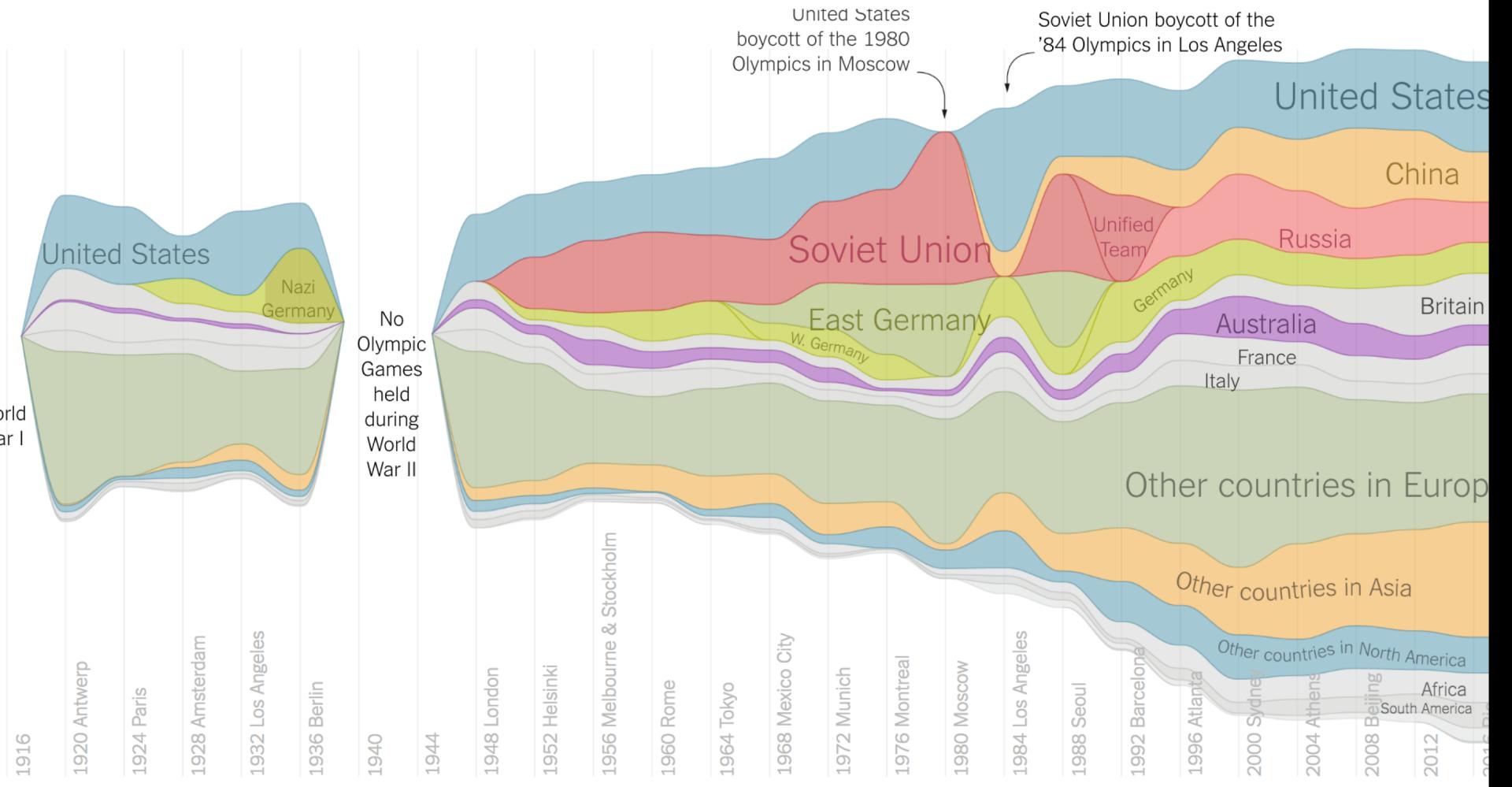


“... a lot remains unknown about how big data-driven decisions may or may not use factors that are proxies for race, sex, or other traits that U.S. laws generally prohibit from being used in a wide range of commercial decisions ... What can be done to make sure these products and services—and the companies that use them treat consumers fairly and ethically?”

- FTC Commissioner Julie Brill [2015]



OLYMPIC MEDALS



NETFLIX PRIZE I

Recommender systems: predict a user's rating of an item

	Twilight	Wall-E	Twilight II	Furious 7
User 1	+1	-1	+1	?
User 2	+1	-1	?	?
User 3	-1	+1	-1	+1

Netflix Prize: \$1MM to the first team that beats our in-house engine by 10%

- Happened after about three years
- Model was **never used** by Netflix for a variety of reasons
 - Out of date (DVDs vs streaming)
 - Too complicated / not interpretable

NETFLIX PRIZE II

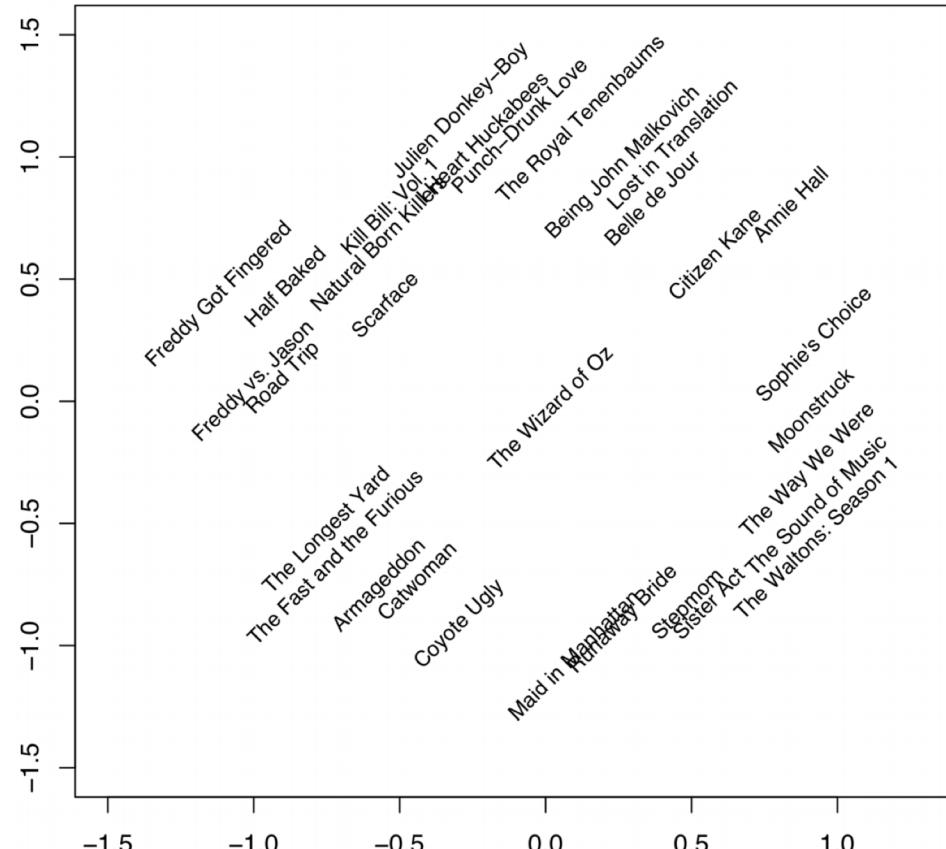
Frat/Gross-Out Comedy

Critically-Acclaimed/Strong Female Lead

Artsy

Latent factor 1

BB



Latent factors model:

Identify factors with max discrimination between movies

NETFLIX PRIZE III

Netflix initially planned a follow-up competition

In 2007, UT Austin managed to deanonymize portions of the original released (anonymized) Netflix dataset:

- ??????????????
- Matched rating against those made publicly on IMDb

Why could this be bad?

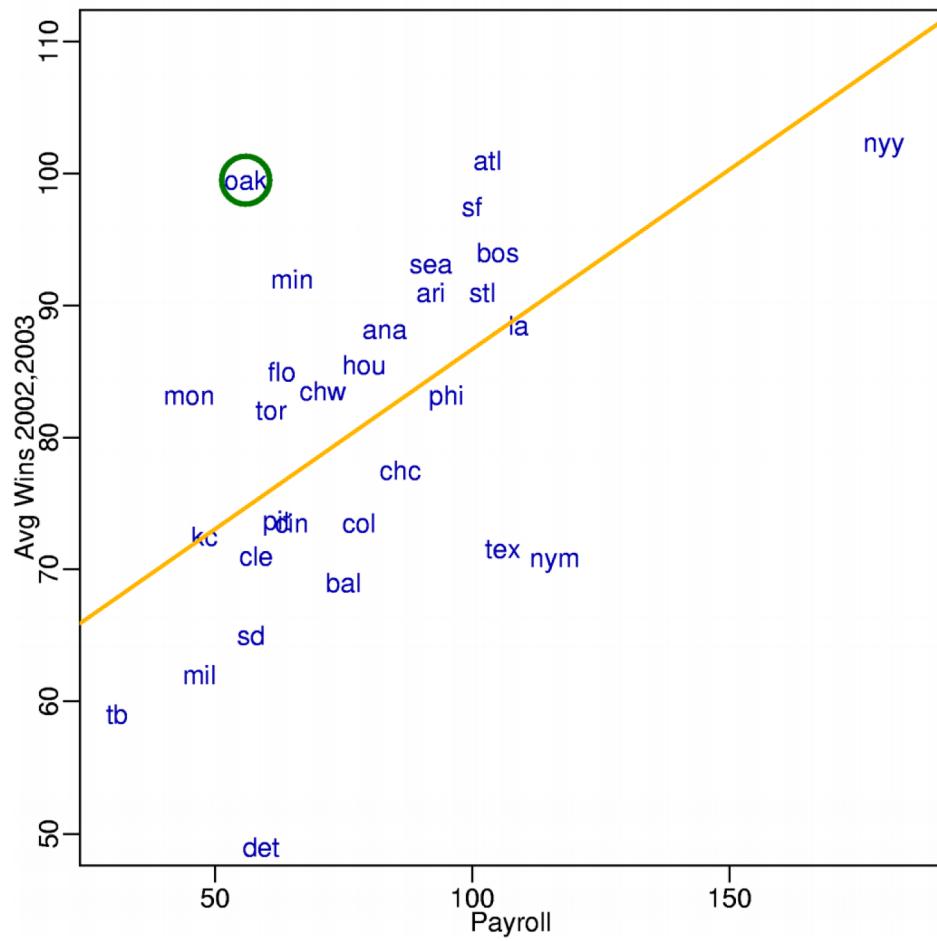
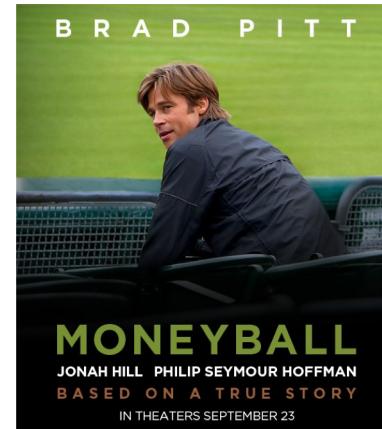
2009—2010, four Netflix users filed a class-action lawsuit against Netflix over

MONEYBALL

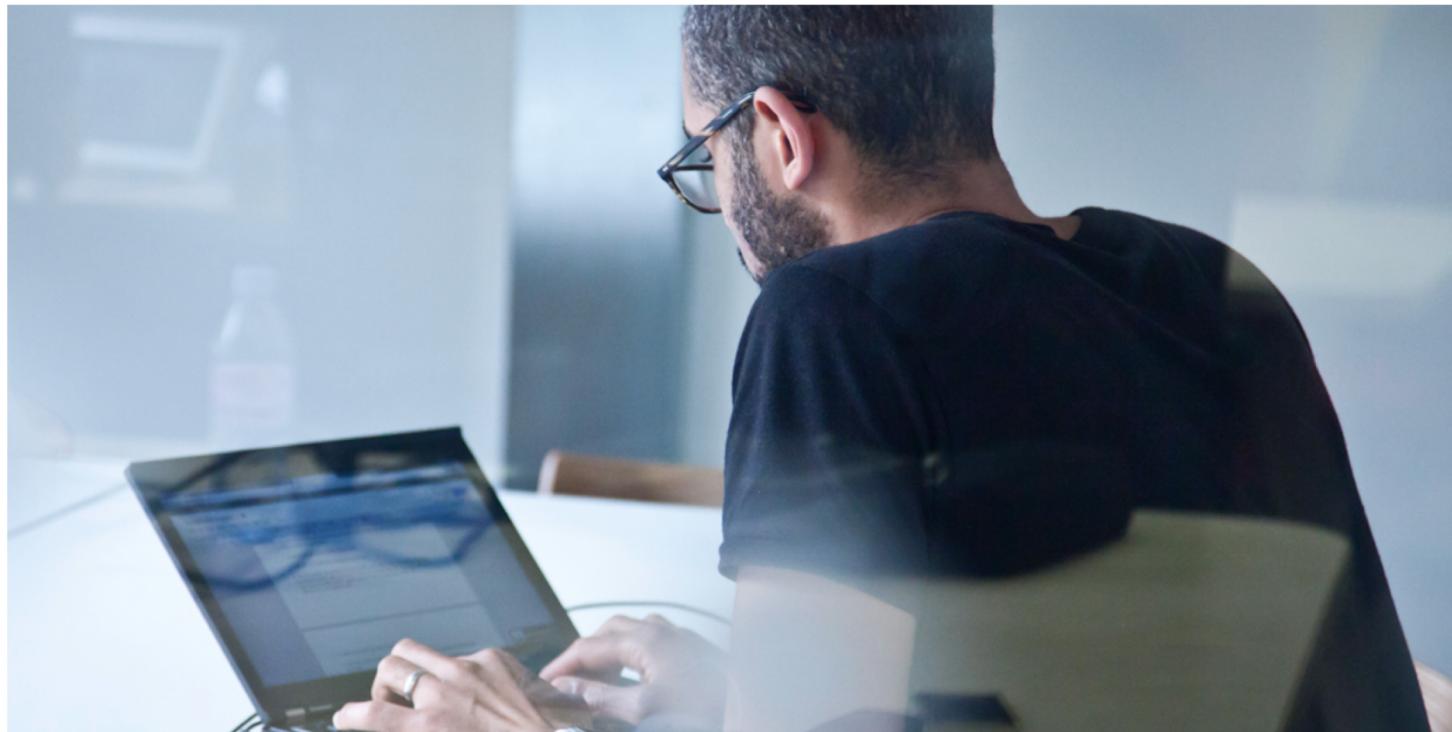
Baseball teams drafted rookie players primarily based on human scouts' opinions of their talents

Peter Brand, data scientist *du jour*, convinces the {bad, poor} Oakland Athletics to use a quantitative aka sabermetric approach to hiring

(Spoiler: Red Sox offer Brand a job, he says no, they take a sabermetric approach and win the World Series.)



1. Data scientist



Shutterstock

Overall job score (out of 5.0): 4.8

Job satisfaction rating (out of 5.0): 4.4

Number of job openings: 4,184

Median base pay: \$110,000

<http://www.businessinsider.com/best-jobs-in-america-in-2017-2017-1/>

WRAP-UP FOR PART I

Register on Piazza using your UMD address:

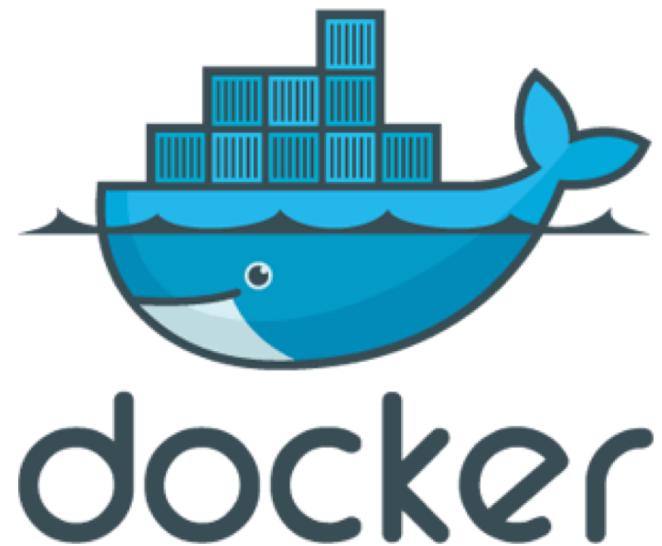
piazza.com/umd/fall2018/cmsc641

Please chat with me if you're unsure of whether or not you're at the right {programming, math} level for this course:

- My guess is that you are!
- This is a young class, so we're quite flexible

Read about Docker & Jupyter!

- Works on *nix, OSX, Windows
- <https://www.docker.com/>
- (We'll post a small project shortly.)



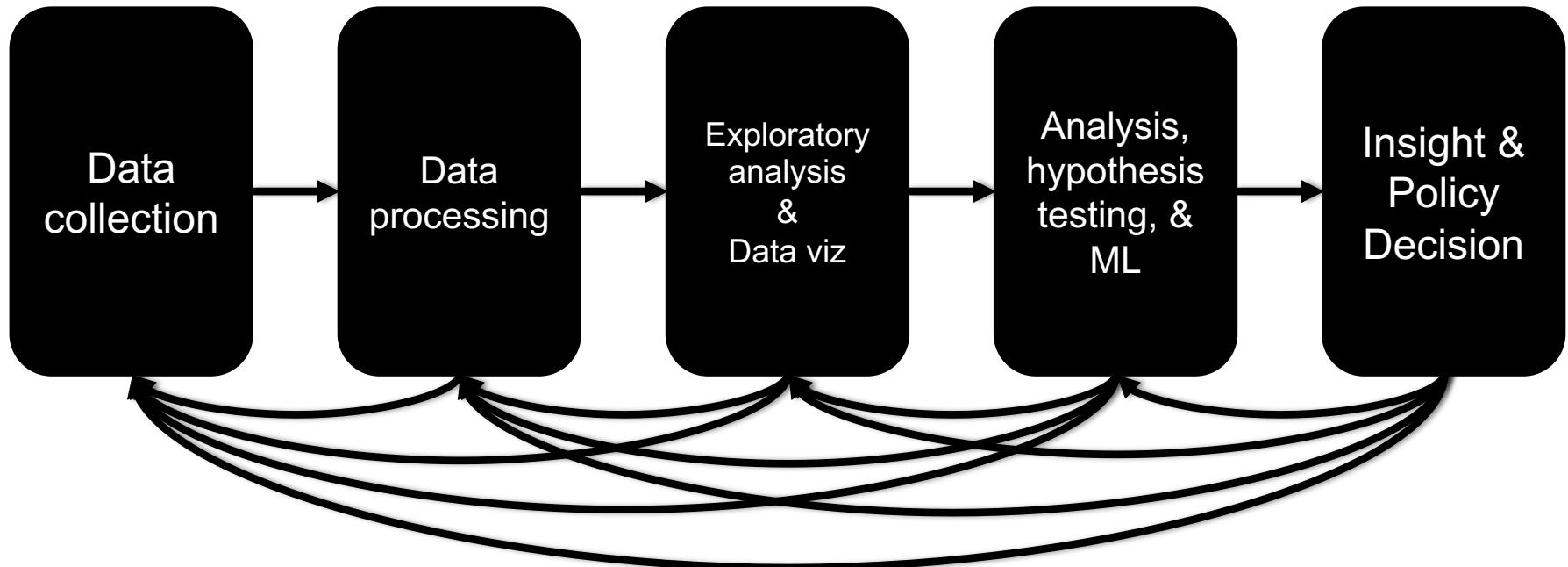


AFTER THE BREAK:

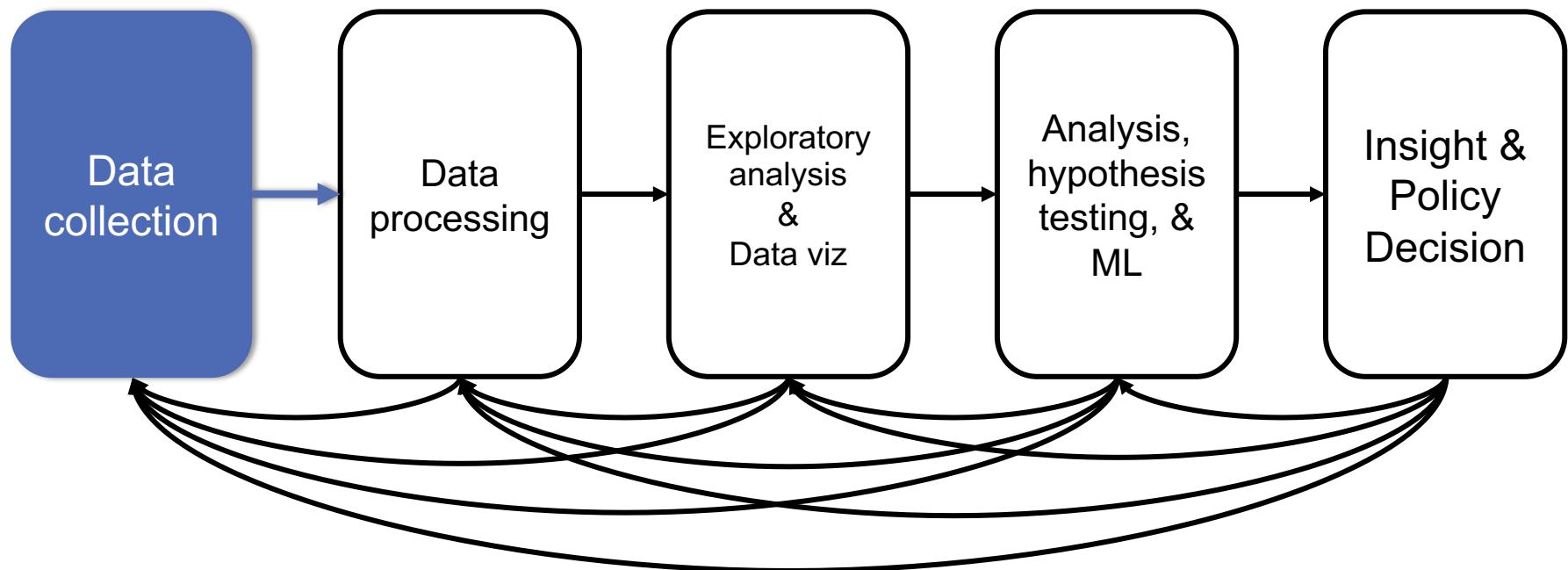
SCRAPING DATA WITH PYTHON



THE DATA LIFECYCLE



TODAY'S LECTURE





BUT FIRST, SNAKES!

Python is an interpreted, dynamically-typed, high-level, garbage-collected, object-oriented-functional-imperative, and widely used scripting language.

- **Interpreted:** instructions executed without being compiled into (virtual) machine instructions*
- **Dynamically-typed:** verifies type safety at runtime
- **High-level:** abstracted away from the raw metal and kernel
- **Garbage-collected:** memory management is automated
- **OOFI:** you can do bits of OO, F, and I programming

Not the point of this class!

- Python is **fast** (developer time), **intuitive**, and **used in industry!**

*you can compile Python source, but it's not required

THE ZEN OF PYTHON

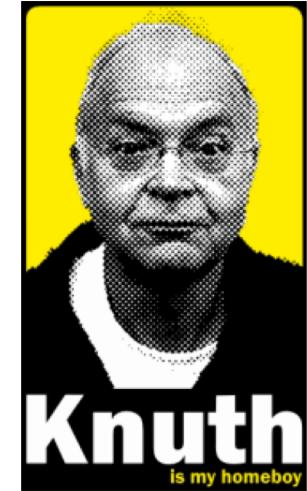
- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules ...
 - ... although practicality beats purity.
- Errors should never pass silently ...
 - ... unless explicitly silenced.



LITERATE PROGRAMMING

Literate code contains in **one document**:

- the **source** code;
- text **explanation** of the code; and
- the **end result** of running the code.



Basic idea: present code in the order that logic and flow of human thoughts demand, not the machine-needed ordering

- Necessary for data science!
- Many choices made need textual explanation, ditto results.

Stuff you'll be using in Project 0 (and beyond)!

IP[y]: IPython

Interactive Computing



jupyter

10-MINUTE PYTHON PRIMER

Define a function:

```
def my_func(x, y):  
    if x > y:  
        return x  
    else:  
        return y
```

Python is whitespace-delimited

Define a function that returns a tuple:

```
def my_func(x, y):  
    return (x-1, y+2)  
  
(a, b) = my_func(1, 2)
```

a = 0; b = 4

USEFUL BUILT-IN FUNCTIONS: COUNTING AND ITERATING

len: returns the number of items of an enumerable object

```
len( [ 'c', 'm', 's', 'c', 3, 2, 0] )
```

```
7
```

range: returns an iterable object

```
list( range(10) )
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

enumerate: returns iterable tuple (index, element) of a list

```
enumerate( ["311", "320", "330"] )
```

```
[(0, "311"), (1, "320"), (2, "330")]
```

<https://docs.python.org/3/library/functions.html>

USEFUL BUILT-IN FUNCTIONS: MAP AND FILTER

map: apply a function to a sequence or iterable

```
arr = [1, 2, 3, 4, 5]
map(lambda x: x**2, arr)
```

```
[1, 4, 9, 16, 25]
```

filter: returns a list of elements for which a predicate is true

```
arr = [1, 2, 3, 4, 5, 6, 7]
filter(lambda x: x % 2 == 0, arr)
```

```
[2, 4, 6]
```

We'll go over in much greater depth with pandas/numpy.

PYTHONIC PROGRAMMING

Basic iteration over an array in Java:

```
int[] arr = new int[10];
for(int idx=0; idx<arr.length; ++idx) {
    System.out.println( arr[idx] );
}
```

Direct translation into Python:

```
idx = 0
while idx < len(arr):
    print( arr[idx] ); idx += 1
```

A more “Pythonic” way of iterating:

```
for element in arr:
    print( element )
```

LIST COMPREHENSIONS

Construct sets like a mathematician!

- $P = \{ 1, 2, 4, 8, 16, \dots, 2^{16} \}$
- $E = \{ x \mid x \text{ in } \mathbb{N} \text{ and } x \text{ is odd and } x < 1000 \}$

Construct lists like a mathematician **who codes!**

```
P = [ 2**x for x in range(17) ]
```

```
E = [ x for x in range(1000) if x % 2 != 0 ]
```

Very similar to `map`, but:

- You'll see these way more than `map` in the wild
- Many people consider `map/filter` not "pythonic"
- They can perform differently (`map` is "lazier")

follow
your



EXCEPTIONS

Syntactically correct statement throws an exception:

- tweepy (Python Twitter API) returns “Rate limit exceeded”
- sqlite (a file-based database) returns IntegrityError

```
print('Python', python_version())

try:
    cause_a_NameError
except NameError as err:
    print(err, '-> some extra text')
```

PYTHON 2 VS 3

Python 3 is intentionally backwards incompatible

- (But not *that* incompatible)

Biggest changes that matter for us:

- `print "statement"` → `print("function")`
- `1/2 = 0` → `1/2 = 0.5` and `1//2 = 0`
- ASCII `str` default → default Unicode

Namespace ambiguity fixed:

```
i = 1  
[i for i in range(5)]  
print(i) # ????????
```

TO ANY CURMUDGEONS ...

If you're going to use Python 2 anyway, use the `_future_` module:

- Python 3 introduces features that will throw runtime errors in Python 2 (e.g., with statements)
- `_future_` module incrementally brings 3 functionality into 2
- https://docs.python.org/2/library/_future_.html

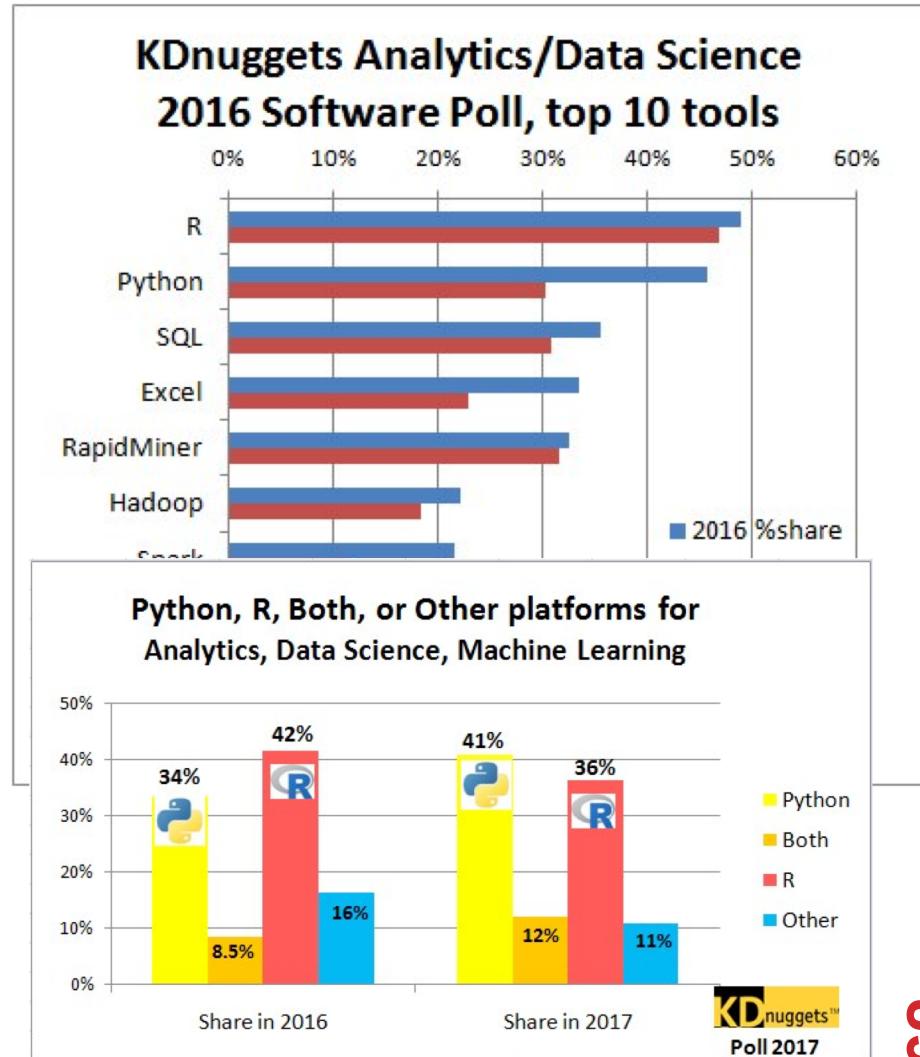
```
from __future__ import division  
from __future__ import print_function  
from __future__ import please_just_use_python_3
```

PYTHON VS R (FOR DATA SCIENTISTS)

There is no right answer here!

- Python is a “full” programming language – easier to integrate with systems in the field
- R has a more mature set of pure stats libraries ...
- ... but Python is catching up quickly ...
- ... and is already ahead specifically for ML.

You will see Python more in the tech industry.



EXTRA RESOURCES

Plenty of tutorials on the web:

- <https://www.learnpython.org/>

Work through Project 0, which will take you through some baby steps with Python and the Pandas library:

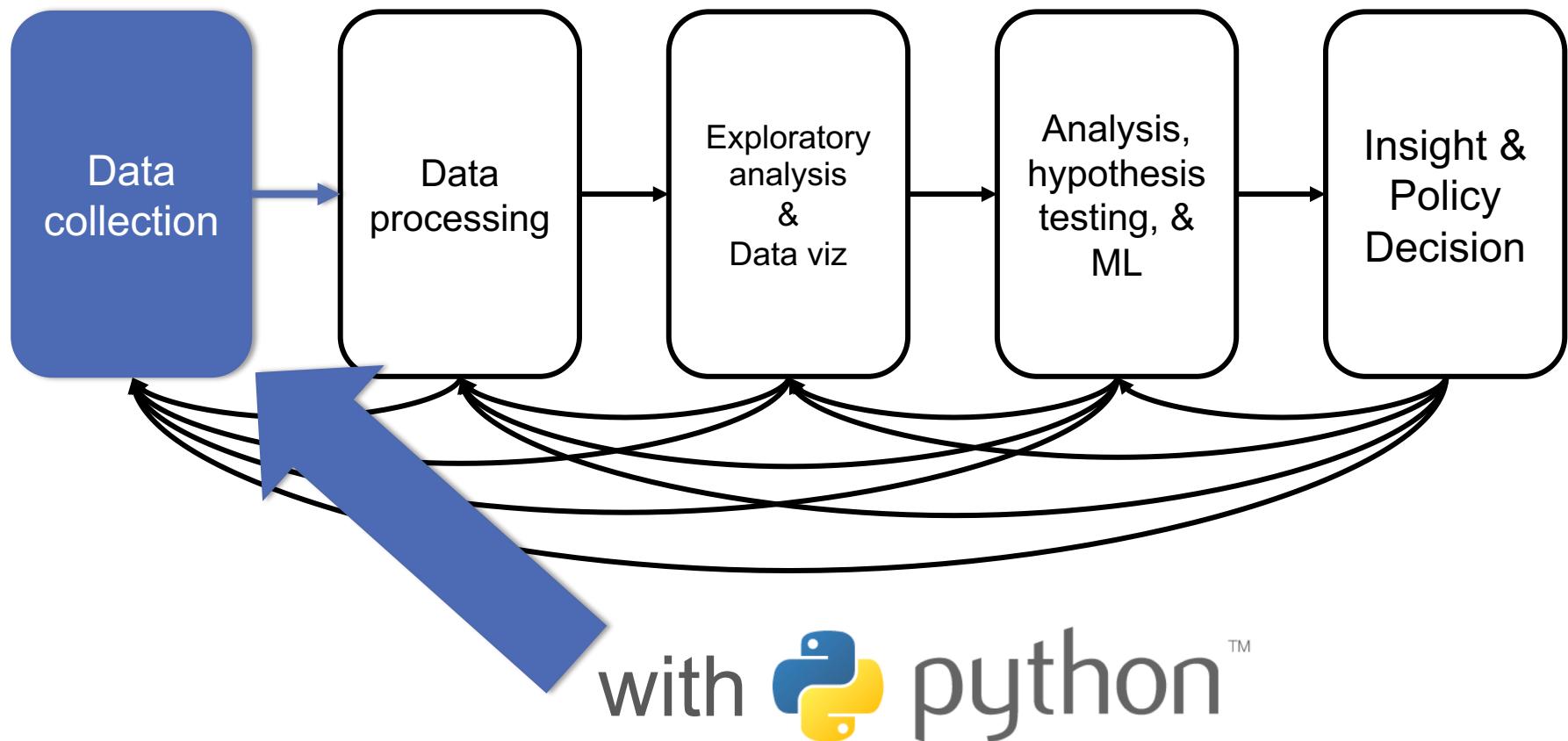
- (We'll also post some readings soon.)

Come hang out at office hours (or chat with me privately)

- Office hours will be on the website/Piazza very soon.
- Also, email me – I realize your schedules are not like undergrads' schedules ☺.



TODAY'S LECTURE



GOTTA CATCH 'EM ALL



Five ways to get data:

- Direct download and load from local storage
- Generate locally via downloaded code (e.g., simulation)
- Query data from a database (covered in a few lectures)
- Query an API from the intra/internet
- Scrape data from a webpage



Covered today.

WHEREFORE ART THOU, API?

A web-based Application Programming Interface (API) like we'll be using in this class is a contract between a server and a user stating:

“If you send me a specific request, I will return some information in a structured and documented format.”

(More generally, APIs can also perform actions, may not be web-based, be a set of protocols for communicating between processes, between an application and an OS, etc.)

“SEND ME A SPECIFIC REQUEST”

Most web API queries we'll be doing will use HTTP requests:

- conda install -c anaconda requests=2.12.4

```
r = requests.get('https://api.github.com/user',
                  auth=('user', 'pass'))
```

```
r.status_code
```

```
200
```

```
r.headers['content-type']
```

```
'application/json; charset=utf8'
```

```
r.json()
```

```
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

HTTP REQUESTS

`https://www.google.com/?q=cmsc320&tbs=qdr:m`



???????????

HTTP GET Request:

GET /?q=cmsc320&tbs=qdr:m HTTP/1.1

Host: www.google.com

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:10.0.1) Gecko/20100101 Firefox/10.0.1

```
params = { "q": "cmsc320", "tbs": "qdr:m" }
r = requests.get(    "https://www.google.com",
                    params = params )
```

*be careful with https:// calls; requests will not verify SSL by default

RESTFUL APIs

This class will just **query** web APIs, but full web APIs typically allow more.

Representational **S**tate **T**ransfer (**R****E****S****T**ful) APIs:

- **GET**: perform query, return data
- **POST**: create a new entry or object
- **PUT**: update an existing entry or object
- **DELETE**: delete an existing entry or object

Can be more intricate, but verbs (“put”) align with actions



QUERYING A RESTFUL API

Stateless: with every request, you send along a token/authentication of who you are

```
token = "super_secret_token"
r = requests.get("https://github.com/user",
                  params={"access_token": token})
print( r.content )
```

```
{"login":"JohnDickerson","id":472985,"avatar_url":"ht...
```

GitHub is more than a GETHub:

- PUT/POST/DELETE can edit your repositories, etc.
- Try it out: <https://github.com/settings/tokens/new>

AUTHENTICATION AND OAUTH

Old and busted:

```
r = requests.get("https://api.github.com/user",
                  auth=("JohnDickerson", "ILoveKittens"))
```

New hotness:

- What if I wanted to grant an app access to, e.g., my Facebook account **without** giving that app my password?
- OAuth: grants **access tokens** that give (possibly incomplete) access to a user or app without exposing a password

“... I WILL RETURN INFORMATION IN A STRUCTURED FORMAT.”

So we've queried a server using a well-formed GET request via the `requests` Python module. What comes back?

General structured data:

- Comma-Separated Value (CSV) files & strings
- Javascript Object Notation (JSON) files & strings
- HTML, XHTML, XML files & strings

Domain-specific structured data:

- Shapefiles: geospatial vector data (OpenStreetMap)
- RVT files: architectural planning (Autodesk Revit)
- You can make up your own! **Always document it.**

CSV FILES IN PYTHON

Any CSV reader worth anything can parse files with any delimiter, not just a comma (e.g., “TSV” for tab-separated)

1,26-Jan,Introduction,—,"pdf, pptx",Dickerson,
2,31-Jan,Scraping Data with Python,Anaconda's Test Drive.,,Dickerson,
3,2-Feb,"Vectors, Matrices, and Dataframes",Introduction to pandas.,,Dickerson,
4,7-Feb,Jupyter notebook lab,,,,"Denis, Anant, & Neil",
5,9-Feb,Best Practices for Data Science Projects,,,Dickerson,

Don't write your own CSV or JSON parser

```
import csv
with open("schedule.csv", "rb") as f:
    reader = csv.reader(f, delimiter=",", quotechar='''')
    for row in reader:
        print(row)
```

(We'll use pandas to do this much more easily and efficiently)

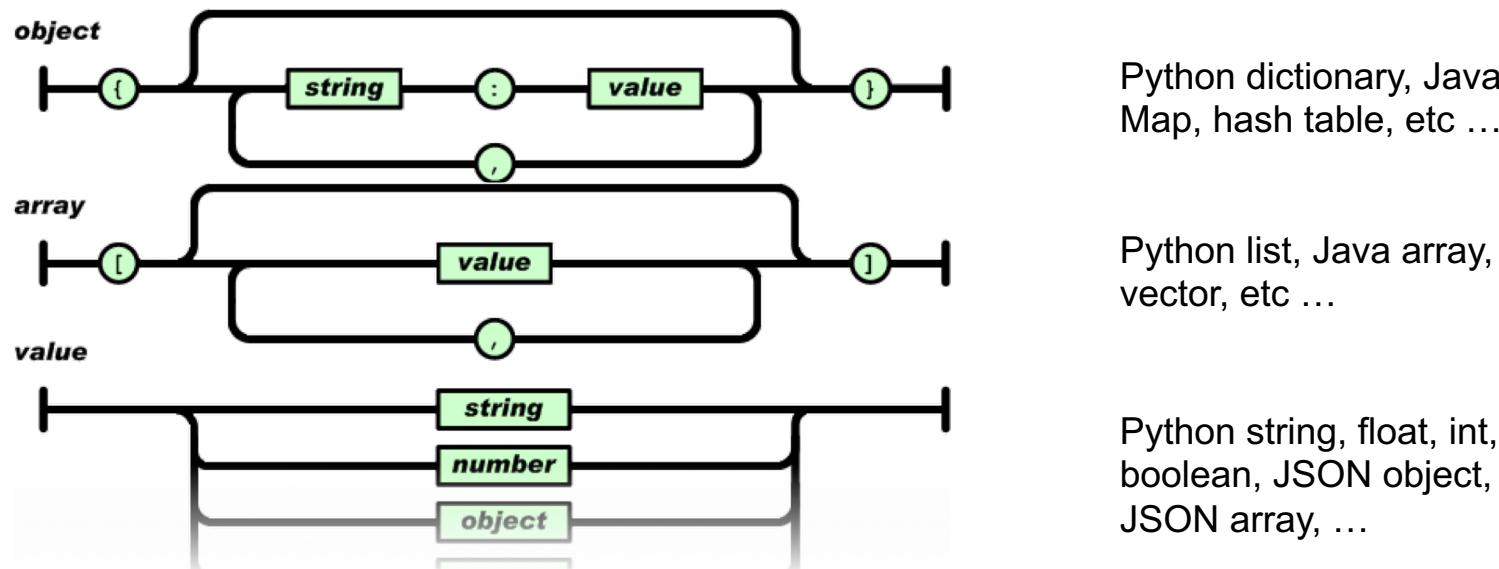
JSON FILES & STRINGS

JSON is a method for **serializing** objects:

- Convert an object into a string (done in Java in 131/132?)
- **Deserialization** converts a string back to an object

Easy for humans to read (and sanity check, edit)

Defined by three universal data structures



JSON IN PYTHON

Some built-in types: “Strings”, 1.0, True, False, None

Lists: [“Goodbye”, “Cruel”, “World”]

Dictionaries: {“hello”: “bonjour”, “goodbye”, “au revoir”}

Dictionaries within lists within dictionaries within lists:

```
[1, 2, {"Help": [  
    "I'm", {"trapped": "in"},  
    "CMSC641"  
]}]
```



JSON FROM TWITTER

```
GET https://api.twitter.com/1.1/friends/list.json?cursor=-1&screen_name=twitterapi&skip_status=true&include_user_entities=false
```

```
{
  "previous_cursor": 0,
  "previous_cursor_str": "0",
  "next_cursor": 1333504313713126852,
  "users": [
    {
      "profile_sidebar_fill_color": "252429",
      "profile_sidebar_border_color": "181A1E",
      "profile_background_tile": false,
      "name": "Sylvain Carle",
      "profile_image_url":
      "http://a0.twimg.com/profile_images/2838630046/4b82e286a659fae310012520f4f756bb_normal.png",
      "created_at": "Thu Jan 18 00:10:45 +0000 2007", ...
    }
  ]
}
```

PARSING JSON IN PYTHON

Repeat: don't write your own CSV or JSON parser

- <https://news.ycombinator.com/item?id=7796268>
- rsdy.github.io/posts/dont_write_your_json_parser_plz.html

Python comes with a fine JSON parser

```
import json

r = requests.get(
    "https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=JohnPDickerson&count=100", auth=auth )

data = json.loads(r.content)
```

```
json.load(some_file) # loads JSON from a file
json.dump(json_obj, some_file) # writes JSON to file
json.dumps(json_obj) # returns JSON string
```

XML, XHTML, HTML FILES AND STRINGS

Still hugely popular online, but JSON has essentially replaced XML for:

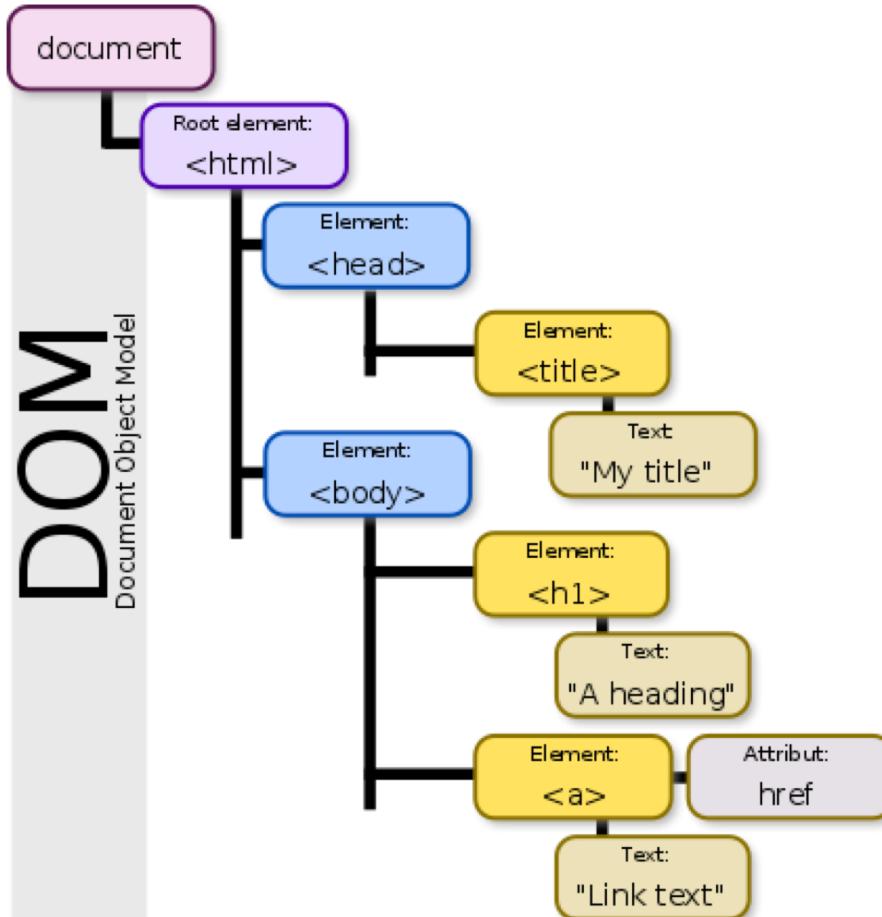
- Asynchronous browser ↔ server calls
- Many (most?) newer web APIs

XML is a hierarchical markup language:

```
<tag attribute="value1">
    <subtag>
        Some cool words or values go here!
    </subtag>
    <openclosetag attribute="value2" />
</tag>
```

You probably won't see much XML, but you will see plenty of HTML, its substantially less well-behaved cousin ...

DOCUMENT OBJECT MODEL (DOM)



SCRAPING HTML IN PYTHON

HTML – the specification – is fairly pure

HTML – what you find on the web – is horrifying

We'll use BeautifulSoup:

- conda install -c asmeurer beautiful-soup=4.3.2



```
import requests
from bs4 import BeautifulSoup

r = requests.get(
    "https://cs.umd.edu/class/spring2017/cmsc320/" )

root = BeautifulSoup( r.content )
root.find("div", id="schedule")\
    .find("table")\                      # find all schedule
    .find("tbody").findAll("a")          # links for CMSC320
```

BUILDING A WEB SCRAPER IN PYTHON

Totally not hypothetical situation:

- You really want to learn about data science, so you choose to download all of last semester's CMSC320 lecture slides to wallpaper your room ...
- ... but you now have carpal tunnel syndrome from clicking refresh on Piazza last night, and can no longer click on the PDF and PPTX links.

Hopeless? No! Earlier, you built a scraper to do this!

```
lnks = root.find("div", id="schedule")\
    .find("table")\                      # find all schedule
    .find("tbody").findAll("a") # links for CMSC320
```

Sort of. You only want PDF and PPTX files, not links to other websites or files.

REGULAR EXPRESSIONS

Given a list of URLs (strings), how do I find only those strings that end in *.pdf or *.pptx?

- Regular expressions!
- (Actually Python strings come with a built-in `endswith` function.)

```
"this_is_a_filename.pdf".endswith((".pdf", ".pptx"))
```

What about .pDf or .pPTx, still legal extensions for PDF/PPTX?

- Regular expressions!
- (Or cheat the system again: built-in string `lower` function.)

```
"tHiS_IS_a_FileNAme.pDF".lower().endswith(  
    (".pdf", ".pptx"))
```

EVERYBODY STAND BACK.



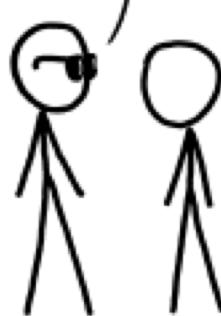
I KNOW REGULAR
EXPRESSIONS.



IF YOU'RE HAVIN' PERL
PROBLEMS I FEEL
BAD FOR YOU, SON-



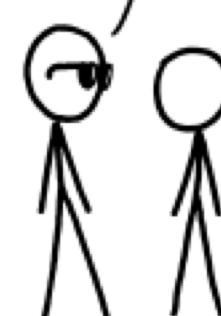
I GOT 99
PROBLEMS,



SO I USED
REGULAR
EXPRESSIONS.



NOW I HAVE
100 PROBLEMS.



REGULAR EXPRESSIONS

Used to **search** for specific elements, or groups of elements, that match a pattern

```
import re

# Find the index of the 1st occurrence of "cmsc641"
match = re.search(r"cmsc641", text)
print( match.start() )
```

```
# Does start of text match "cmsc641"?
match = re.match(r"cmsc641", text)
```

```
# Iterate over all matches for "cmsc641" in text
for match in re.finditer(r"cmsc641", text):
    print( match.start() )
```

```
# Return all matches of "cmsc641" in the text
match = re.findall(r"cmsc641", text)
```

MATCHING MULTIPLE CHARACTERS

Can match sets of characters, or multiple and more elaborate sets and sequences of characters:

- Match the character 'a': a
- Match the character 'a', 'b', or 'c': [abc]
- Match any character except 'a', 'b', or 'c': [^abc]
- Match any digit: \d (= [0123456789] or [0-9])
- Match any alphanumeric: \w (= [a-zA-Z0-9_])
- Match any whitespace: \s (= [\t\n\r\f\v])
- Match any character: .

Special characters must be escaped: .^\$*+?{}\\[]|()

MATCHING SEQUENCES AND REPEATED CHARACTERS

A few common modifiers (available in Python and most other high-level languages; +, {n}, {n,} may not):

- Match character ‘a’ exactly once: a
- Match character ‘a’ zero or once: a?
- Match character ‘a’ zero or more times: a*
- Match character ‘a’ one or more times: a+
- Match character ‘a’ exactly n times: a{ n }
- Match character ‘a’ at least n times: a{ n , }

Example: match all instances of “University of <somewhere>” where <somewhere> is an alphanumeric string with at least 3 characters:

- \s*University\sof\s\w{3, }

COMPILED REGEXES

If you're going to reuse the same regex many times, or if you aren't but things are going slowly for some reason, try **compiling** the regular expression.

- <https://blog.codinghorror.com/to-compile-or-not-to-compile/>

```
# Compile the regular expression "cmsc320"
regex = re.compile(r"cmsc320")

# Use it repeatedly to search for matches in text
regex.match( text )      # does start of text match?
regex.search( text )     # find the first match or None
regex.findall( text )    # find all matches
```

Interested? CMSC6*, CMSC7*, CMSC8*, talk to me.

DOWNLOADING A BUNCH OF FILES

Import the modules

```
import re
import requests
from bs4 import BeautifulSoup
try:
    from urllib.parse import urlparse
except ImportError:
    from urlparse import urlparse
```

Get some HTML via HTTP

```
# HTTP GET request sent to the URL url
r = requests.get( url )

# Use BeautifulSoup to parse the GET response
root = BeautifulSoup( r.content )
lnks = root.find("div", id="schedule") \
    .find("table") \
    .find("tbody").findAll("a")
```

DOWNLOADING A BUNCH OF FILES

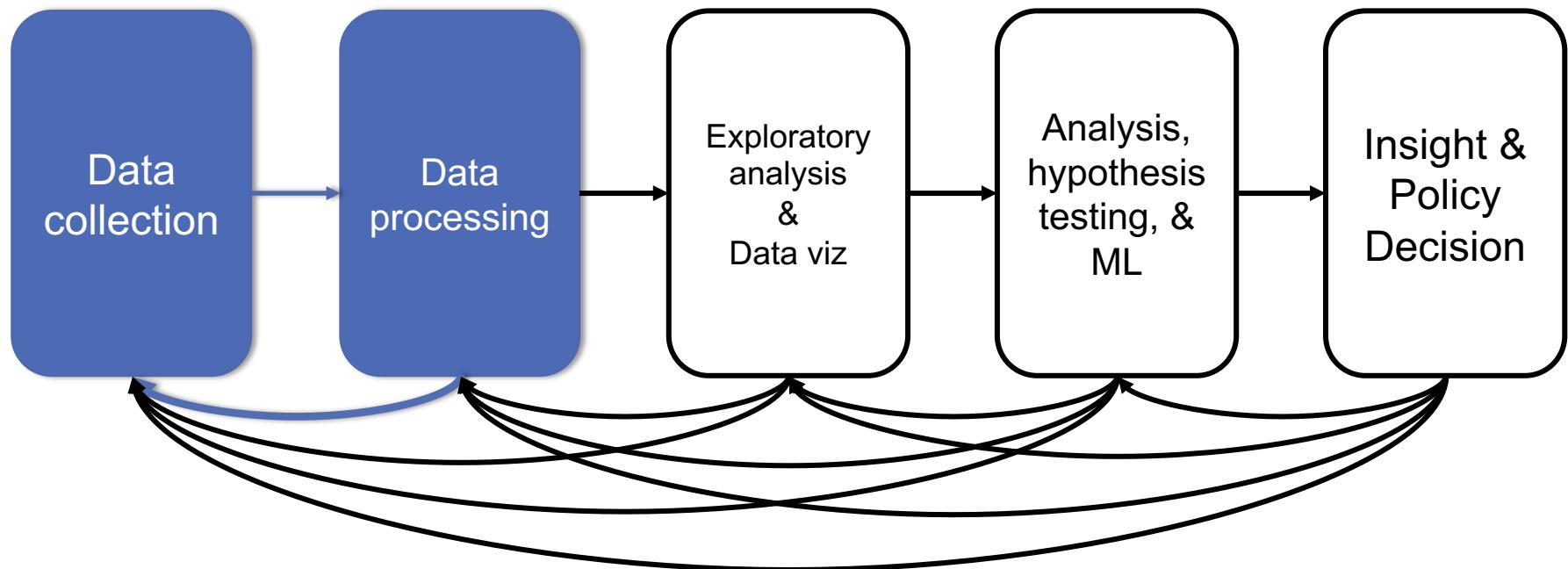
Parse exactly what you want

```
# Cycle through the href for each anchor, checking  
# to see if it's a PDF/PPTX link or not  
for lnk in lnks:  
    href = lnk['href']  
  
    # If it's a PDF/PPTX link, queue a download  
    if href.lower().endswith('.pdf', '.pptx'):
```

Get some more data?!

```
urld = urlparse.urljoin(url, href)  
rd = requests.get(urld, stream=True)  
  
# Write the downloaded PDF to a file  
outfile = path.join(outbase, href)  
with open(outfile, 'wb') as f:  
    f.write(rd.content)
```

NEXT LECTURE





NEXT CLASS:

NUMPY, SCIPY, AND DATAFRAMES

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

