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/](https://pages.github.com/) – make a GitHub account, too!

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

It will be graded 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?

Thanks to: John Dickerson
GRADE BREAKDOWN

40% mini-projects:
• There are 4 of them
• Equal weighting @ 10% each

10% reading homeworks

25% Midterm

25% final tutorial
SOME TECHNOLOGIES WE WILL USE

\[ y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it} \]
(Don’t tell CMSC330 …)
IMPORTANT WALLS OF TEXT
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.
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

in creating a brand.)

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

Hiring

Lending

Search for minority names → ads for DUI/arrest records

Policing/sentencing

Female cookies → less freq. shown professional job opening ads

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

<table>
<thead>
<tr>
<th></th>
<th>Twilight</th>
<th>Wall-E</th>
<th>Twilight II</th>
<th>Furious 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
<td>?</td>
</tr>
<tr>
<td>User 2</td>
<td>+1</td>
<td>-1</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>User 3</td>
<td>-1</td>
<td>+1</td>
<td>-1</td>
<td>+1</td>
</tr>
</tbody>
</table>

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
MONEYBALL

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

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

(Spoiler: Red Sox offer Bean a job, he says no, they take a sabermetric approach and win the World Series.)
Glassdoor, a popular job site, again published a list of 50 Best Jobs in America, and in 2018, for the third year in a row, Data Scientist ranked as the no. 1 job. Data Scientist had an overall Job score 4.8 out of 5, $110,000 Median Base Salary, and over 4,000 job openings on Glassdoor.
WRAP-UP FOR PART I

Register on Piazza using your UMD address:

piazza.com/umd/summer2019/cmsc320

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/
• (Project 0 is posted.)
THE DATA LIFECYCLE

Data collection → Data processing → Exploratory analysis & Data viz → Analysis, hypothesis testing, & ML → Insight & Policy Decision
TODAY’S LECTURE

Data collection

Data processing

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision
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
Define a function:

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

Define a function that returns a tuple:

```python
def my_func(x, y):
    return (x-1, y+2)

(a, b) = my_func(1, 2)
```

```python
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](https://docs.python.org/3/library/functions.html)
USEFUL BUILT-IN FUNCTIONS: MAP AND FILTER

**map**: apply a function to a sequence or iterable

```python
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

```python
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.
Basic iteration over an array in Java:

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

Direct translation into Python:

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

A more “Pythonic” way of iterating:

```python
for element in arr:
    print( element )
```
LIST COMPREHENSIONS

Construct sets like a mathematician!
- \( P = \{ 1, 2, 4, 8, 16, \ldots, 2^{16} \} \)
- \( E = \{ x \mid x \text{ in } \mathbb{N} \text{ and } x \text{ is odd and } x < 1000 \} \)

Construct lists like a mathematician who codes!

\[
P = [ 2**x \text{ for } x \text{ in range}(17) ]
\]

\[
E = [ x \text{ for } x \text{ in range}(1000) \text{ if } x \% 2 \neq 0 ]
\]

- 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”)
SYNTACTICALLY CORRECT STATEMENT THROWS AN EXCEPTION:

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

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

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

```python
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

```python
from _future_ import division
from _future_ import print_function
from _future_ import please_just_use_python_3
```
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
• Office hours are on the website.
• Also, email me – if it doesn’t fit your schedules.
TODAY’S LECTURE

Data collection

Data processing

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Thanks: Zico Kolter’s 15-388
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=2.21.0`

```python
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