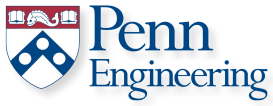
Data *Transformation*

Susan B. Davidson and Zachary G. Ives

University of Pennsylvania

CIS 545 – Big Data Analytics



Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-24-22

We Have Structured Data – Are We There Yet?

https://tinyurl.com/cis545-notebook-01

"Part 2: Data Transformation"

company_data_df

company_ceos_df

execs_df

We have lots of data, but fragmented!

How do we:

(1) Make sure it's encoded the way we want

(2) Stitch the data together?

https://tinyurl.com/cis545-lecture-01-24-22

We Now Know How to Bring Data into Pandas

Populating our Big Data Toolbox:

Basic operations (building blocks) to clean and link tables

As we build towards a solution for our company-CEO question

https://tinyurl.com/cis545-lecture-01-24-22

A Few Remarks on Our Focus

There are *many* libraries, tools, and languages for big data

Almost all of them compile to the **same underlying operations**, which are amenable to scaling up!

We'll study these operations in several settings!

https://tinyurl.com/cis545-lecture-01-24-22

The Main Programming Models We'll Learn

1. **Pandas dataframes** – direct control, not persistent
2. **SQL** – automatically optimized, persistent

3. Soon: **Apache Spark**, which combines all of the above!

Let's start by reviewing a few things we need to do…

https://tinyurl.com/cis545-lecture-01-24-22

company_ceos_df:

	Company	Executive	Title	Since	Notes	Updated
0	Accenture	Julie Sweet	CEO[1]	2019	Succeeded Pierre Nanterme, Passed Away	2019-01-31
1	Aditya Birla Group	Kumar Birla	Chairman[2]	1995[2]	Part of the Birla family business house in India	2018-10-01
2	Being Short	Meghan	Chairman, president and CEO[3]	2007	Formerly with Apple Inc.	2018-10-01

execs_df:

0	Julie_Sweet	https://en.wikipedia.org/wiki/Julie_Sweet	NaT
1	Kumar_Birla	https://en.wikipedia.org/wiki/Kumar_Birla	1967-06-14
2	Shantanu_Narayen	https://en.wikipedia.org/wiki/Shantanu_Narayen	1963-05-27
3	Garo_H._Armen	https://en.wikipedia.org/wiki/Garo_H._Armen	1953-01-31
4	Tom_Enders	https://en.wikipedia.org/wiki/Tom_Enders	NaT
5	Daniel_Zhang	https://en.wikipedia.org/wiki/Daniel_Zhang	1972-01-11
6	Jeff_Bezos	https://en.wikipedia.org/wiki/Jeff_Bezos	1964-01-12

Need to **clean** name

Need to filter missing data

https://tinyurl.com/cis545-lecture-01-24-22

Road Map

- Column-wise operations
- Filtering rows
- Joining or combining tables
  - Leading us to the *record linking* problem for next time
- Ethical principles around data

https://tinyurl.com/cis545-lecture-01-24-22

Columnwise Operations

Susan B. Davidson and Zachary G. Ives
University of Pennsylvania
CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-24-22

Data Cleaning

- Our first task: "clean" the contents of the **executive** column in **company_ceos_df**, which had underscores instead of spaces

- To do this: we need to start with *projection…*

https://tinyurl.com/cis545-lecture-01-24-22

Projecting from a Dataframe

Projection in Pandas has two forms:

```
# Double-brackets: return dataframe
exec_df[['name', 'born']]
```

	name	born
0	Julie_Sweet	NaT
1	Kumar_Birla	1967-06-14
2	Shantanu_Narayen	NaT

```
# Single brackets: 1 column as a Series
exec_df['name']
```

```
0 Julie_Sweet
1 Kumar_Birla
2 Shantanu_Narayen
3 Garo_H._Armen
```

https://tinyurl.com/cis545-lecture-01-24-22

Computing Over a Series with a Function

It's best NOT to iterate over the elements and modify them

- Instead: call **apply** with a function!

```
def to_space(x):
  return x.replace('_', ' ')

# *apply* to each element, returning a new Series
exec_df['name'].apply(to_space)
```

```
0 Julie Sweet
1 Kumar Birla
2 Shantanu Narayen
```

doesn't replace old one!

https://tinyurl.com/cis545-lecture-01-24-22

Computing Over a Series with a Lambda Function

It's best NOT to iterate over the elements and modify them

- Instead: call **apply** with a function!

```
0 Julie Sweet
1 Kumar Birla
2 Shantanu Narayen
```

```
# *apply* to each element, returning a new Series
exec_df['name'].apply(lambda x: x.replace('_', ' '))
```

https://tinyurl.com/cis545-lecture-01-24-22

Computing Over Rows in a Dataframe

```
# *apply* a function over rows of a dataframe
exec_df.apply(lambda x: x['name'].replace('_', ' ')
, axis=1)
```

Many Pandas functions use:

- axis = 0 column at a time
- axis = 1 row at a time

```
0 Julie Sweet
1 Kumar Birla
2 Shantanu Narayen
```

https://tinyurl.com/cis545-lecture-01-24-22

Inserting the Results Back into the Dataframe

```
# Let's clean the name!
exec_df['clean_name'] = exec_df['name'].apply
(lambda x: x.replace('_', ' '))
```

	name	page	born	clean_name
0	Julie_Sweet	https://en.wikipedia.org/wiki/Julie_Sweet	NaT	**Julie Sweet**
1	Kumar_Birla	https://en.wikipedia.org/wiki/Kumar_Birla	1967-06-14	**Kumar Birla**
2	Shantanu_Narayen	https://en.wikipedia.org/wiki/Shantanu_Narayen	NaT	**Shantanu Narayen**

https://tinyurl.com/cis545-lecture-01-24-22

Can I Change Column Names?

```
exec_df.rename(columns={'name': 'old_name'})
```

	old_name	page	born	clean_name
0	Julie_Sweet	https://en.wikipedia.org/wiki/Julie_Sweet	NaT	Julie Sweet
1	Kumar_Birla	https://en.wikipedia.org/wiki/Kumar_Birla	1967-06-14	Kumar Birla
2	Shantanu_Narayen	https://en.wikipedia.org/wiki/Shantanu_Narayen	NaT	Shantanu Narayen
3	Garo_H._Armen	https://en.wikipedia.org/wiki/Garo_H._Armen	1953-01-31	Garo H. Armen
4	Guillaume_Faury	https://en.wikipedia.org/wiki/Guillaume_Faury	1968-02-22	Guillaume Faury

https://tinyurl.com/cis545-lecture-01-24-22

Why Not Iteration?

- Iterating over items in a loop forces a sequence
- This basic model of projection -> apply is *possible to do in parallel*
- We'll see how to do so with various Python and other tools!

https://tinyurl.com/cis545-lecture-01-24-22

Another Variation of the Same Idea

- Sometimes our dataframe is in a database on disk – using SQL we can fetch *just the portions we want* and do our computations…

```
select col1, col2 AS new_name1, expr1 AS new_name2
from table
```

https://tinyurl.com/cis545-lecture-01-24-22

What if the Data is in a Database?

Dataframes must fit in memory… but what if the data is too large, or stored in a database?

```
exec_df.to_sql('exec', conn, if_exists="replace")

-----------------------------------------------------------------

pd.read_sql_query('select name, replace(name, "_", " ") as
clean_name from exec', conn)
```

	name	clean_name
0	Julie_Sweet	Julie Sweet
1	Kumar_Birla	Kumar Birla
2	Meghan	Meghan

https://tinyurl.com/cis545-lecture-01-24-22

Quick Review

https://canvas.upenn.edu/courses/1636888/quizzes/2779127 (03B)

1. How do `my_df['col1']` and `my_df[['col1']]` differ?
   a. The first returns a series and the second returns a dataframe
   b. The first is correct and the second is a typo
   c. They are the same
   - The first returns a dataframe and the second returns a series
3. A lambda function is:
   a. A function that doesn't return any values
   b. A function that takes no parameters
   c. An unnamed function that's defined where it's to be used, which takes input parameters and returns a result
   d. A function defined with the name lambda
4. Iteration is undesirable over big data because:
   a. It requires for loops
   b. It sequentializes operations instead of letting Python decide how to do the evaluation
   c. It only works over columns in a dataframe

https://tinyurl.com/cis545-lecture-01-24-22

The Story So Far

- We've seen how to extract single columns and subsets of columns via *projection*
- The *apply* operation allows us to do computation over a field or the contents of a row
- We can assign the results back to columns in a dataframe

Next: what about filtering "bad" rows?

https://tinyurl.com/cis545-lecture-fall22

Filtering Rows

Susan B. Davidson and Zachary G. Ives

University of Pennsylvania

CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-24-22

Selecting Items

We treat rows very differently from columns:

Columns are *properties* – a name, a date, etc.

`dataframe[['col1','col2']]`

A row represents a sample or *instance*

`dataframe` rows with particular values

https://tinyurl.com/cis545-lecture-01-24-22

Selecting a Row by a Name

Suppose we want to find the URL for

`clean_name` = Julie Sweet…

In Pandas, we need to do this by first

defining a *series* of True / False (Boolean) values

https://tinyurl.com/cis545-lecture-01-24-22

Selecting Only *Some* of the Rows

```
3   exec_df['clean_name']
```

```
0              Julie Sweet
1              Kumar Birla
2         Shantanu Narayen
3            Garo H. Armen
4          Guillaume Faury
```

...

https://tinyurl.com/cis545-lecture-01-24-22

Selecting Only *Some* of the Rows

```
[3]: exec_df['clean_name']
```

```
0              Julie Sweet
1              Kumar Birla
2         Shantanu Narayen
3            Garo H. Armen
4          Guillaume Faury
...
```

```
[3]: exec_df['clean_name'] == 'Julie Sweet'
```

```
0     True
1    False
2    False
3    False
4    False
...
```

Filter "mask" – each row is given a `True` or `False`, we only return those with `True`

https://tinyurl.com/cis545-lecture-01-24-22

Selecting Only *Some* of the Rows

```
3  exec_df['clean_name']
```

```
0          Julie Sweet
1          Kumar Birla
2      Shantanu Narayen
3         Garo H. Armen
4      Guillaume Faury
```

```
3  exec_df['clean_name'] == 'Julie Sweet'
```

```
0     True
1    False
2    False
3    False
4    False
```

```
3  exec_df[exec_df['clean_name'] == 'Julie Sweet']
```

	name	page	born	clean_name
0	Julie_Sweet	https://en.wikipedia.org/wiki/Julie_Sweet	NaT	Julie Sweet

https://tinyurl.com/cis545-lecture-01-24-22

Selecting Only *Some* of the Rows … And the Page URL

```
3  exec_df['clean_name']
```

```
0          Julie Sweet
1          Kumar Birla
2       Shantanu Narayen
3          Garo H. Armen
4       Guillaume Faury
```

```
3  exec_df['clean_name'] == 'Julie Sweet'
```

```
0     True
1    False
2    False
3    False
4    False
```

```
1  exec_df[exec_df['clean_name'] == 'Julie Sweet'][['page']]
```

	page
0	https://en.wikipedia.org/wiki/Julie_Sweet

https://tinyurl.com/cis545-lecture-01-24-22

Note the *Composition* of Steps

- Define a "filter mask" – a Boolean Series with True or False for each row
- Use that to request a subset of the rows
- Then project a column!

We can do the same in SQL, in one operation…

https://tinyurl.com/cis545-lecture-01-24-22

Selection in SQL…

```
3  exec_df.to_sql('temp_exec', conn, if_exists="replace")
4
5  pd.read_sql_query('select * from temp_exec where clean_name="Julie Sweet"', conn)
```

```
1  pd.read_sql_query('select page from temp_exec where clean_name="Julie Sweet"', conn)
```

	page
0	https://en.wikipedia.org/wiki/Julie_Sweet

https://tinyurl.com/cis545-lecture-01-24-22

Can We Filter People with Missing Birthdays?

```
1  import numpy as np
2
3  exec_df.dropna(subset=['born'])
```

```
1  pd.read_sql_query('select * from temp_exec where born is not null', conn).set_index('index')
```

index	name	page	born
1	Kumar_Birla	https://en.wikipedia.org/wiki/Kumar_Birla	1967-06-14 00:00:00
3	Garo_H._Armen	https://en.wikipedia.org/wiki/Garo_H._Armen	1953-01-31 00:00:00
4	Guillaume_Faury	https://en.wikipedia.org/wiki/Guillaume_Faury	1968-02-22 00:00:00

https://tinyurl.com/cis545-lecture-01-24-22

Brief Review

https://canvas.upenn.edu/courses/1636888/quizzes (03C)

1. What does `my_df['col1'] > 3` return?
   a. A series of Boolean values, true only for rows where `col1` is greater than 3
   b. A series of `col1` values from each row in the dataframe
   c. A subset of rows in `my_df` where `col1` is greater than 3
   d. It generates a syntax error
2. Which SQL statement is equivalent to `my_df[my_df['a'] != 3]`?
   a. `select * from my_df where my_df['a'] != 3`
   b. `select * from my_df`
   c. `select * from my_df where a <> 3`
   d. `select my_df where a = 3`

https://tinyurl.com/cis545-lecture-01-24-22

Now We Have a Basic Set of Operations Over (Single) Tables

- Projection – pull out a "slice" of the table on certain columns
- Selection – pull out rows matching conditions
- **apply()** to compute over columns
- In-place updates for Dataframes

- Basic SQL: **select** *columns, expressions* **from** *table* **where** *conditions*

- Next: let's look at combining tables!

https://tinyurl.com/cis545-lecture-01-24-22

Joining Tables

Susan B. Davidson and Zachary G. Ives

University of Pennsylvania

CIS 545 – Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-24-22

Can We Put the Data Together?

company_ceos_df

execs_df

- We now want to "connect" the data – we can do this via a **merge** aka a **join**, which matches rows with the same value

```
exec_df[['clean_name', 'born']]
```

clean_name	born
Kumar Birla	1967-06-14 00:00:00
Garo H. Armen	1953-01-31 00:00:00
Daniel Zhang	1972-01-11 00:00:00

```
company_ceos_df[['Executive', 'Company']]
```

Executive	Company
David Rowland	Accenture
Kumar Birla	Aditya Birla Group
Shantanu Narayen	Adobe Systems
Garo H. Armen	Agenus

https://tinyurl.com/cis545-lecture-01-24-22

The Join (Merge) Operation

```
1  company_ceos_df[['Executive', 'Company']].merge(exec_df[['clean_name', 'born']],
2                                                  left_on=['Executive']
3                                                  right_on=['clean_name'])
```

	Executive	Company	clean_name	born
0	Kumar Birla	Aditya Birla Group	Kumar Birla	1967-06-14 00:00:00
1	Garo H. Armen	Agenus	Garo H. Armen	1953-01-31 00:00:00
2	Guillaume Faury	Airbus	Guillaume Faury	1968-02-22 00:00:00
3	Daniel Zhang	Alibaba	Daniel Zhang	1972-01-11 00:00:00
4	Jeff Bezos	Amazon.com	Jeff Bezos	1964-01-12 00:00:00
5	Lisa Su	AMD	Lisa Su	1969-11-07 00:00:00

https://tinyurl.com/cis545-lecture-01-24-22

In SQL

```
company_ceos_df.to_sql('company_ceos', conn, if_exists="replace")
exec_df.to_sql('executives', conn, if_exists="replace")

pd.read_sql_query('select Executive, Company, born from company_ceos '
+\
                  'join executives on Executive=clean_name', conn)
```

Executive	Company	born
Kumar Birla	Aditya Birla Group	1967-06-14 00:00:00
Garo H. Armen	Agenus	1953-01-31 00:00:00
Guillaume Faury	Airbus	1968-02-22 00:00:00
Daniel Zhang	Alibaba	1972-01-11 00:00:00
Jeff Bezos	Amazon.com	1964-01-12 00:00:00
Lisa Su	AMD	1969-11-07 00:00:00

https://tinyurl.com/cis545-lecture-01-24-22

An Issue!

```
company_ceos_df
[['Executive', 'Company']]
```

166	Herbert_Diess
169	Stefano_Pessina
170	Doug_McMillon
176	Vince_McMahon

```
exec_df[['clean_name', 'born']]
```

166	Herbert Diess	1958-10-24
169	Stefano Pessina	1941-06-04
170	Doug McMillon	1966-10-17
176	Vince McMahon	1945-08-24

Join Output

173	Greg Creed	Yum! Brands	Greg Creed	NaT
174	Rich Barton	Zillow Group	Rich Barton	NaT

https://tinyurl.com/cis545-lecture-01-24-22

Let's Try the Join Again
As a *Left Outerjoin* with an *Indicator*

```
company_ceos_df[['Executive', 'Company']].merge(exec_df[['clean_name',
'born']],  left_on=['Executive'],  right_on=['clean_name'],
                    how="left", indicator=True)
```

	Executive	Company	clean_name	born	_merge
0	Julie Sweet	Accenture	Julie Sweet	NaT	both
1	Kumar Birla	Aditya Birla Group	Kumar Birla	1967-06-14	both
2	Shantanu Narayen	Adobe Systems	Shantanu Narayen	NaT	both
3	Garo H. Armen	Agenus	Garo H. Armen	1953-01-31	both
4	Guillaume Faury	Airbus	Guillaume Faury	1968-02-22	both
24	Harald Krüger	BMW	NaN	NaN	left_only

https://tinyurl.com/cis545-lecture-01-24-22

More Generally

```
result_df = cor
            mer

result_df[resul
```

	Executive	Company	clean_name	born	_merge
24	Harald Krüger	BMW	NaN	NaT	left_only
41	Ola Källenius	Daimler AG	NaN	NaT	left_only
51	Börje Ekholm	Ericsson	NaN	NaT	left_only
127	Michael O'Leary	Ryanair	NaN	NaT	left_only
179	NaN	NaN	Harald Kr%C3%BCger	1965-10-13	right_only
180	NaN	NaN	Ola K%C3%A4llenius	NaT	right_only
181	NaN	NaN	B%C3%B6rje Ekholm	NaT	right_only
182	NaN	NaN	Michael O%27Leary	NaT	right_only

https://tinyurl.com/cis545-lecture-01-24-22

Exact-Match (Inner) Joins vs Outerjoins

Joins allow us to match on sub-fields

- By default, and in Pandas, they are on **equality only**

Outerjoin will include "partial" rows when one side (e.g., the left) doesn't have a match on the other side (e.g., the right)

Sometimes we can "lose" results without noticing – motivates our next module on **data cleaning** and **record linking**

https://tinyurl.com/cis545-lecture-01-24-22

Composition!
Joining a Joined Result

```
1  pd.read_sql_query('select Executive, Company, born  from company_ceos ' +\
2                    'join executives on Executive=clean_name join company_data cd on Company=cd.name ' +\
3                    'where born is not null', conn)
```

	Executive	Company	born
0	Garo H. Armen	Agenus	1953-01-31 00:00:00
1	Daniel Zhang	Alibaba	1972-01-11 00:00:00
2	Brian L. Roberts	Comcast	1959-06-28 00:00:00
3	Mark Zuckerberg	Facebook	1984-05-14 00:00:00
9	Jack Dorsey	Twitter	1976-11-19 00:00:00
10	Hans Vestberg	Verizon Communications	1965-06-23 00:00:00

name is ambiguous, so we need to give a *table variable* to company_data

note we only have 10 matches!!! outerjoin?

https://tinyurl.com/cis545-lecture-01-24-22

Other Issues

```
pd.read_sql_query('select * from company_ceos ' +\
                  'join executives on Executive=clean_name '  +\
            'join company_data cd on Company=cd.name ' +\
                  'where born is not null', conn)
```

	index	Company	Executive	Title	Since	Notes	Updated	index	name	page
0	3	Agenus	Garo H. Armen	Founder, chairman, CEO[4]	1994	Founder of the Children of Armenia Fund (COAF)	2018-10-01	3	Garo_H._Armen	https://en.wikipedia.org/wiki/Garo_H._Armer
1	5	Alibaba	Daniel Zhang	CEO[6]	2015	Previously with Taobao	2018-10-01	5	Daniel_Zhang	https://en.wikipedia.org/wiki/Daniel_Zhang

2 indexes, 2 names, etc.!

https://tinyurl.com/cis545-lecture-01-24-22

In Pandas, _x and _y are Added

```
company_ceos_df.merge(exec_df, left_on=['Executive'],
                      right_on=['clean_name']).\
                      merge(company_data_df,
                            left_on='Company',right_on='name')
```

	Company	Executive	Title	Since	Notes	Updated	name_x	name_y
0	Agenus	Garo H. Armen	Founder, chairman, CEO[4]	1994	Founder of the Children of Armenia Fund (COAF)	2018-10-01	Garo_H._Armen	Agenus
1	Alibaba	Daniel Zhang	CEO[6]	2015	Previously with Taobao	2018-10-01	Daniel_Zhang	Alibaba
2	Comcast	Brian L. Roberts	Chairman and CEO[38]	2002	Son of Ralph J. Roberts, the founder of Comcast	2017-11-14	Brian_L._Roberts	Comcast

https://tinyurl.com/cis545-lecture-01-24-22

Quick Review

https://canvas.upenn.edu/courses/1636888/quizzes (03D)

1. Which is true?
   a. An outerjoin produces a subset of the results of the inner join
   b. An outerjoin always has more results than an inner join
   c. An outerjoin will return the exact same results as an innerjoin if every tuple in the input relations has a match.
   d. A left outerjoin returns a subset of a right outerjoin
2. If we join relations that each have an id column, what happens in Pandas?
   a. You will get an exception from Pandas
   b. The input columns will be renamed in the output, to be unique, by appending suffixes to the names
   c. The output will only have one of the ids
   d. You will get a Python compilation error

https://tinyurl.com/cis545-lecture-01-24-22

Recap: Joins

- In Pandas: merge combines rows from two tables **if** they exactly match on column values
- In SQL we can specify a more general condition for the join
- Outerjoins are the same as (inner) joins EXCEPT when there's no match for a tuple
- We can compose joins to link multiple tables

https://tinyurl.com/cis545-lecture-01-24-22

Data Science Ethics: Data Acquisition

Susan B. Davidson and Zachary G. Ives
University of Pennsylvania
CIS 545: Big Data Analytics

Portions of this lecture have been contributed to the OpenDS4All project, piloted by Penn, IBM, and the Linux Foundation

https://tinyurl.com/cis545-lecture-01-24-22

Data Science Models Affect Every Aspect of Society

- Admission to schools
- Who to hire (and who to fire)
- Work schedules
- Whether to grant a loan
- What ads are shown, discounts are given
- What news and social media posts you see
- ….

https://tinyurl.com/cis545-lecture-01-24-22

A good data scientist needs to understand the ethical issues surrounding the data they obtain/use, the algorithms they employ, and its impact on people.

https://tinyurl.com/cis545-lecture-01-24-22

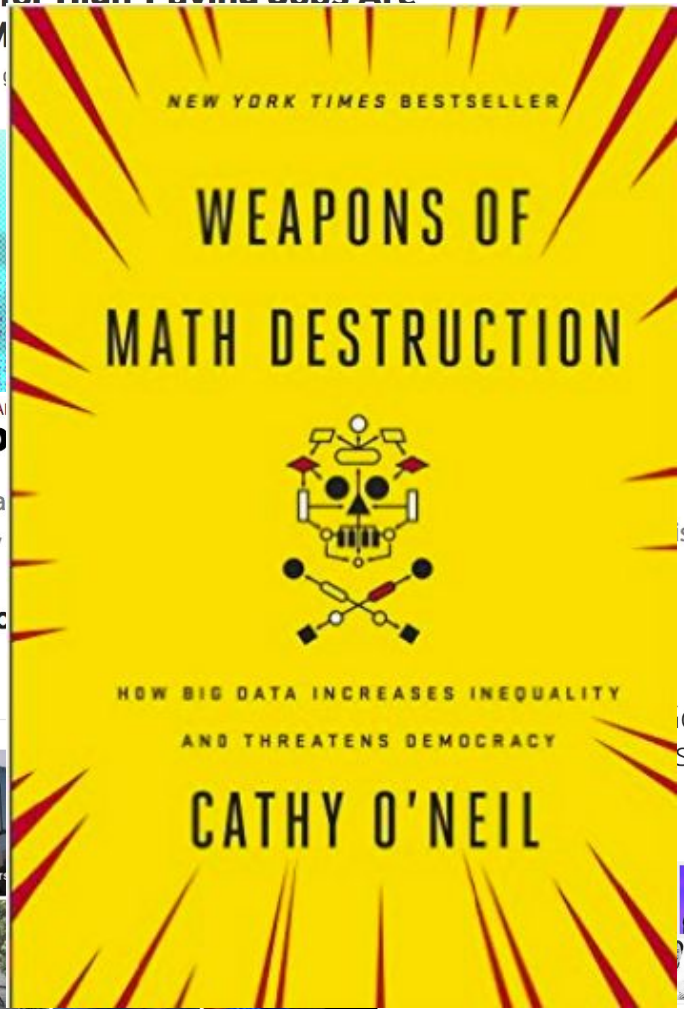


IMAGE: PERCENTAGE OF WOMEN IN TOP 100 GOOGLE IMAGE SEARCH RESULTS FOR CEO IS: 11 PERCENT. PERCENTAGE OF US CEOS WHO ARE WOMEN IS: 27 PERCENT. view more ›

dson, L. Ungar

What are Ethics?

- Rules that we all voluntarily follow because it makes the world a better place.

- Not following rules may have negative consequences
  - Legal consequences
  - Non-legal consequences

https://tinyurl.com/cis545-lecture-01-24-22

Ethical Principles for Human Subject Research

- **Respect for persons**
  - Informed consent
- **Beneficence**
  - Maximization of possible benefits to society and subjects
  - Minimization of harm to subjects
- **Justice**
  - Fairness in distribution of research benefits and burdens

https://tinyurl.com/cis545-lecture-01-24-22

Ethical Principles around Data

Ownership

- The right to control your own data, possibly via surrogates

Consent

- You should understand how your data will be used, and explicitly approve its use

Privacy

- As your data is used, all reasonable efforts should be made to protect privacy.

Openness

- Data should be freely available and have no restrictions

https://tinyurl.com/cis545-lecture-01-24-22

https://en.wikipedia.org/wiki/Big_data_ethics
Image courtesy of HHS.gov

OKCupid Data Publicly Released

May 8, 2016: Danish researchers publicly released a dataset of ~70,000 OKcupid users

- Usernames, age, gender, location, what kind of relationship they are interested in
- Personality traits, answers to 1000s of profiling questions

Why didn't they attempt to anonymize?

- Researcher's response: "… all the data found in the dataset are or were already publicly available, so releasing this dataset merely presents it in a more useful form."

https://tinyurl.com/cis545-lecture-01-24-22

WIRED, Michael Zimmer 5/14/16

Was the OKCupid Data "Public"?

The data was acquired by screen scraping – however, the exact methodology was not fully explained

OKCupid users may restrict the visibility of their profiles to *logged-in users only*

- Likely that the researchers collected—and released—profiles that were intended *not* to be publicly viewable

So this is a violation of privacy.

https://tinyurl.com/cis545-lecture-01-24-22

Data Acquisition: Some Considerations

- Is the data publicly available? Or do you have to create an account or use a login to access the data?

- Do you give appropriate attribution?
  - A “citation” to the .csv and .json files used in the first in-class exercise is given in the README.md file in ./sample_data
  - A “citation” to the data used in the lecture notebooks is the url to where the data can be found

- Is scraping Web data ethical?
  - Could we publish the data about CEOs and their birthdays scraped from Wikipedia?

https://tinyurl.com/cis545-lecture-01-24-22

Closing Thoughts

A good data scientist needs to understand the ethical issues surrounding the data they obtain/use, the algorithms they employ, and its impact on people.

https://tinyurl.com/cis545-lecture-01-24-22

Brief Review

https://canvas.upenn.edu/courses/1636888/quizzes (03E)

1. In 2016, researchers publicly released a dataset of ~70,000 OKCupid users. Which of the following ethic principles of data were **not** violated?

   a. Consent

   b. Privacy

   c. Ownership

   d. Openness

https://tinyurl.com/cis545-lecture-01-24-22

Next Time

How do we address two open issues:

Detecting and **cleaning** errors in the data

Matching / joining in a more tolerant way?

https://tinyurl.com/cis545-lecture-01-24-22