Data Comes From Many Sources

Bob Jacobsen, UC Berkeley

Merging data - Drinks

```
In [2]:  # create a table of drinks available at several places with there prices
         drinks = Table(['Drink', 'Cafe', 'Price']).with_rows([  # a table of menus for cafes
             ['Milk Tea', 'Tea One',  4],
             ['Espresso', 'Nefeli',   2],
             ['Latte',    'Nefeli',   3],
             ['Espresso', "Abe's",    2]
         ])
         drinks
```

Out[2]:

Drink	Cafe	Price
Milk Tea	Tea One	4
Espresso	Nefeli	2
Latte	Nefeli	3
Espresso	Abe's	2

```
In [3]:  # create a table of available discounts
         discounts = Table().with_columns(                 # A table of discounts by cafe
             'Coupon % off', make_array(25, 50, 5),
             'Location', make_array('Tea One', 'Nefeli', 'Tea One')
         )
         discounts
```

Out[3]:

Coupon % off	Location
25	Tea One
50	Nefeli
5	Tea One

Bob Jacobsen, UC Berkeley

```
In [4]: # combine the tables by matching cafe names
        t = drinks.join('Cafe', discounts, 'Location')
        t               # note you don't have a discount for Abe's
```

Out[4]:

Cafe	Drink	Price	Coupon % off
Nefeli	Espresso	2	50
Nefeli	Latte	3	50
Tea One	Milk Tea	4	25
Tea One	Milk Tea	4	5

```
In [5]: # Compute a column of discounted price
        t.with_column('Discounted', t.column(2) * (1 - t.column(3)/ 100))
```

Out[5]:

Cafe	Drink	Price	Coupon % off	Discounted
Nefeli	Espresso	2	50	1
Nefeli	Latte	3	50	1.5
Tea One	Milk Tea	4	25	3
Tea One	Milk Tea	4	5	3.8

Bob Jacobsen, UC Berkeley

```
In [6]:  # What do all possible two-drink orders cost?
         # Join with itself, matching on Cafe (you only order in one place)
         two = drinks.join('Cafe', drinks)
         two
```

Out[6]:

Cafe	Drink	Price	Drink_2	Price_2
Abe's	Espresso	2	Espresso	2
Nefeli	Espresso	2	Espresso	2
Nefeli	Espresso	2	Latte	3
Nefeli	Latte	3	Espresso	2
Nefeli	Latte	3	Latte	3
Tea One	Milk Tea	4	Milk Tea	4

```
In [7]:  # Add a total price
         two.with_column('Total', two.column('Price') + two.column('Price_2'))
```

Out[7]:

Cafe	Drink	Price	Drink_2	Price_2	Total
Abe's	Espresso	2	Espresso	2	4
Nefeli	Espresso	2	Espresso	2	4
Nefeli	Espresso	2	Latte	3	5
Nefeli	Latte	3	Espresso	2	5
Nefeli	Latte	3	Latte	3	6
Tea One	Milk Tea	4	Milk Tea	4	8

Bob Jacobsen, UC Berkeley

Another way to understand data - GIS - Bikes

CSC
School of Computing

```
In [1]: # usual imports
        from datascience import *
        import numpy as np
        import pandas as pd

        %matplotlib inline
        import matplotlib.pyplot as plots
        #plots.style.use('fivethirtyeight')

        # Configure for presentation
        #np.set_printoptions(threshold=50, linewidth=50)
        import matplotlib as mpl
        #mpl.rc('font', size=16)
```

Bikes

```
In [2]: # Read a dataset from a bike-rental firm containing 354k rentals
        trips = Table.read_table('trip.csv')
        # see what columns are available in this data set:
        trips
```

Out[2]:

| Trip ID	Duration	Start Date	Start Station	Start Terminal	End Date	End Station	End Terminal	Bike #	Subscriber Type	Zip Code
913460	765	8/31/2015 23:26	Harry Bridges Plaza (Ferry Building)	50	8/31/2015 23:39	San Francisco Caltrain (Townsend at 4th)	70	288	Subscriber	2139
913459	1036	8/31/2015 23:11	San Antonio Shopping Center	31	8/31/2015 23:28	Mountain View City Hall	27	35	Subscriber	95032
913455	307	8/31/2015 23:13	Post at Kearny	47	8/31/2015 23:18	2nd at South Park	64	468	Subscriber	94107
913454	409	8/31/2015 23:10	San Jose City Hall	10	8/31/2015 23:17	San Salvador at 1st	8	68	Subscriber	95113
913453	789	8/31/2015 23:09	Embarcadero at Folsom	51	8/31/2015 23:22	Embarcadero at Sansome	60	487	Customer	9069
913452	293	8/31/2015 23:07	Yerba Buena Center of the Arts (3rd @ Howard)	68	8/31/2015 23:12	San Francisco Caltrain (Townsend at 4th)	70	538	Subscriber	94118
913451	896	8/31/2015 23:07	Embarcadero at Folsom	51	8/31/2015 23:22	Embarcadero at Sansome	60	363	Customer	92562
913450	255	8/31/2015	Embarcadero at Sansome	60	8/31/2015	Steuart at Market	74	470	Subscriber	94111

```
In [3]: # identify a subsample of "commuters"
        commute = trips.where('Duration', are.below(1800))   # Why is this here?  Are there significant ones above that?
        commute.hist('Duration')
```
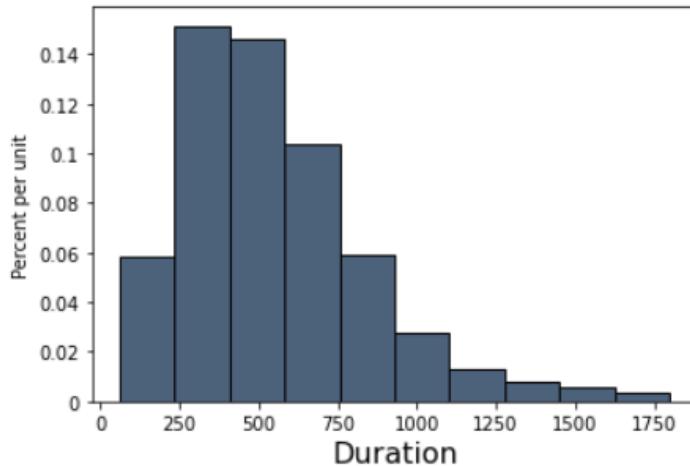


```
In [4]: commute.hist('Duration', bins=60, unit='second')  # clean the plot up a bit
```
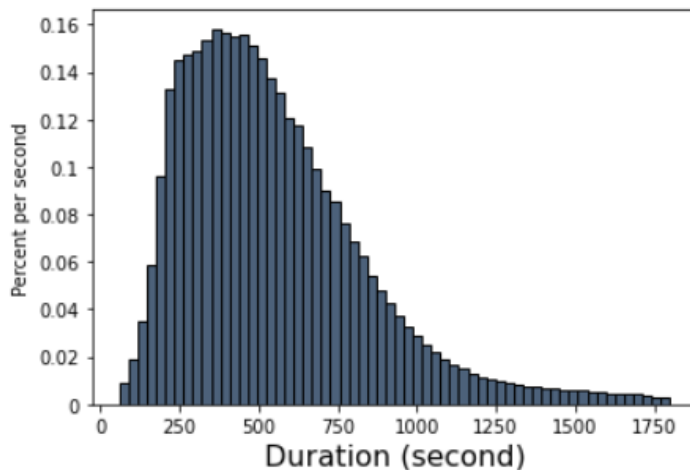
```
In [5]: commute.hist('Duration', bins=np.arange(1801), unit='second')  # there are 354K rows
```
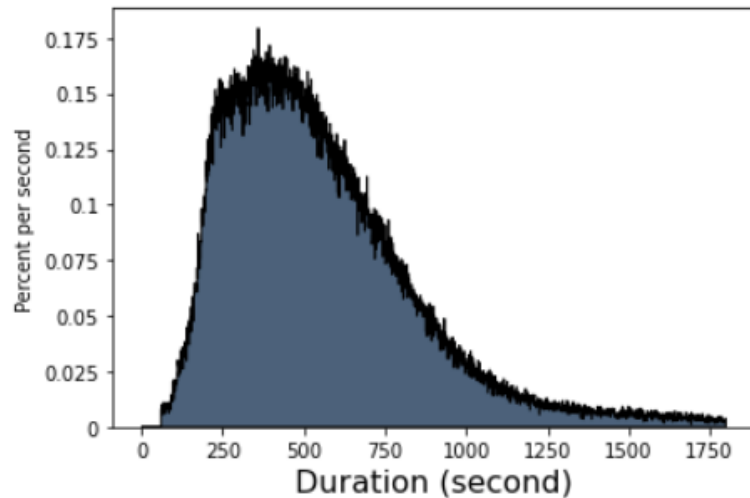


```
In [6]: # group by starting location to get counts, then sort to get largest values
        starts = commute.group('Start Station').sort('count', descending=True)
        starts
```

Out[6]:

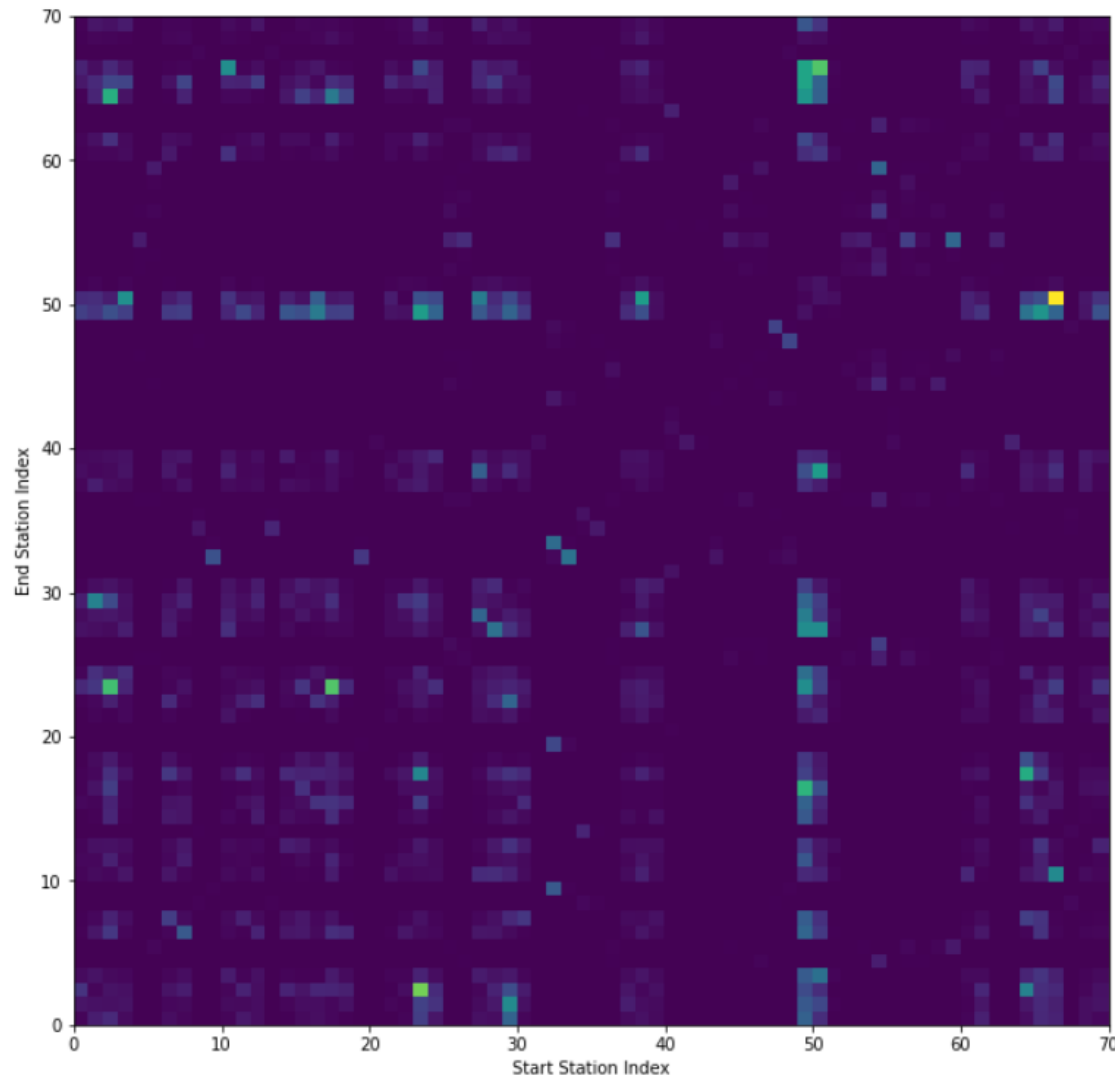| Start Station	count
San Francisco Caltrain (Townsend at 4th)	25858
San Francisco Caltrain 2 (330 Townsend)	21523
Harry Bridges Plaza (Ferry Building)	15543
Temporary Transbay Terminal (Howard at Beale)	14298
2nd at Townsend	13674
Townsend at 7th	13579
Steuart at Market	13215
Embarcadero at Sansome	12842
Market at 10th	11523
Market at Sansome	11023

... (60 rows omitted)

```
In [7]: # Compute a table counting start -> end trips
        pivot = commute.pivot('Start Station', 'End Station')
        pivot
```

Out[7]:

End Station	2nd at Folsom	2nd at South Park	2nd at Townsend	5th at Howard	Adobe on Almaden	Arena Green / SAP Center	Beale at Market	Broadway St at Battery St	California Ave Caltrain Station	Castro Street and El Camino Real	Civic Center BART (7th at Market)	Clay at Battery	Commercial at Montgomery	Ur
2nd at Folsom	54	190	554	107	0	0	40	21	0	0	44	78	54	
2nd at South Park	295	164	71	180	0	0	208	85	0	0	112	87	160	
2nd at Townsend	437	151	185	92	0	0	608	350	0	0	80	329	168	
5th at Howard	113	177	148	83	0	0	59	130	0	0	203	76	129	
Adobe on Almaden	0	0	0	0	11	4	0	0	0	0	0	0	0	
Arena Green / SAP Center	0	0	0	0	7	64	0	0	0	0	0	0	0	
Beale at Market	127	79	183	59	0	0	59	661	0	0	201	75	101	
Broadway St at Battery St	67	89	279	119	0	0	1022	110	0	0	62	283	226	
California Ave Caltrain Station	0	0	0	0	0	0	0	0	38	1	0	0	0	
Castro Street and El Camino Real	0	0	0	0	0	0	0	0	0	30	0	0	0	

```
In [8]:  # It's easier to interpret this graphically — the heat plot
         plots.rcParams['figure.figsize'] = (11., 11.)  # make a square plot
         plots.figure()
         ct = pd.crosstab(commute['Start Station'], commute['End Station']) # pandas computation of pivot table
         plots.grid(False)
         plots.pcolor(ct)     # plot that dataframe as color spectrum
         plots.xlabel('Start Station Index')
         plots.ylabel('End Station Index')
         plots.plot();
```


9

```
In [10]: duration = trips.select('Start Station', 'End Station', 'Duration')  # narrow down the table to three columns
         duration
```

Out[10]:

Start Station	End Station	Duration
Harry Bridges Plaza (Ferry Building)	San Francisco Caltrain (Townsend at 4th)	765
San Antonio Shopping Center	Mountain View City Hall	1036
Post at Kearny	2nd at South Park	307
San Jose City Hall	San Salvador at 1st	409
Embarcadero at Folsom	Embarcadero at Sansome	789
Yerba Buena Center of the Arts (3rd @ Howard)	San Francisco Caltrain (Townsend at 4th)	293
Embarcadero at Folsom	Embarcadero at Sansome	896
Embarcadero at Sansome	Steuart at Market	255
Beale at Market	Temporary Transbay Terminal (Howard at Beale)	126
Post at Kearny	South Van Ness at Market	932

... (354142 rows omitted)

```
In [11]: # Group the trips from each to each, then select the shortest duration trip in each bin
         shortest = duration.group(['Start Station', 'End Station'], min)
         shortest
```

Out[11]:

Start Station	End Station	Duration min
2nd at Folsom	2nd at Folsom	61
2nd at Folsom	2nd at South Park	61
2nd at Folsom	2nd at Townsend	137
2nd at Folsom	5th at Howard	215
2nd at Folsom	Beale at Market	219
2nd at Folsom	Broadway St at Battery St	351

Bob Jacobsen, UC Berkeley

Maps

```
In [13]:  # Get the locations of the stations
          stations = Table.read_table('station.csv')   # Table of station locations
          stations                                     # landmark is the town containg the station
```

Out[13]:

station_id	name	lat	long	dockcount	landmark	installation
2	San Jose Diridon Caltrain Station	37.3297	-121.902	27	San Jose	8/6/2013
3	San Jose Civic Center	37.3307	-121.889	15	San Jose	8/5/2013
4	Santa Clara at Almaden	37.334	-121.895	11	San Jose	8/6/2013
5	Adobe on Almaden	37.3314	-121.893	19	San Jose	8/5/2013
6	San Pedro Square	37.3367	-121.894	15	San Jose	8/7/2013
7	Paseo de San Antonio	37.3338	-121.887	15	San Jose	8/7/2013
8	San Salvador at 1st	37.3302	-121.886	15	San Jose	8/5/2013
9	Japantown	37.3487	-121.895	15	San Jose	8/5/2013
10	San Jose City Hall	37.3374	-121.887	15	San Jose	8/6/2013
11	MLK Library	37.3359	-121.886	19	San Jose	8/6/2013

... (60 rows omitted)

Bob Jacobsen, UC Berkeley

```
In [14]:  # Map all the locations
          Marker.map_table(stations.select('lat', 'long', 'name'))
```

Out[14]:

Bob Jacobsen, UC Berkeley
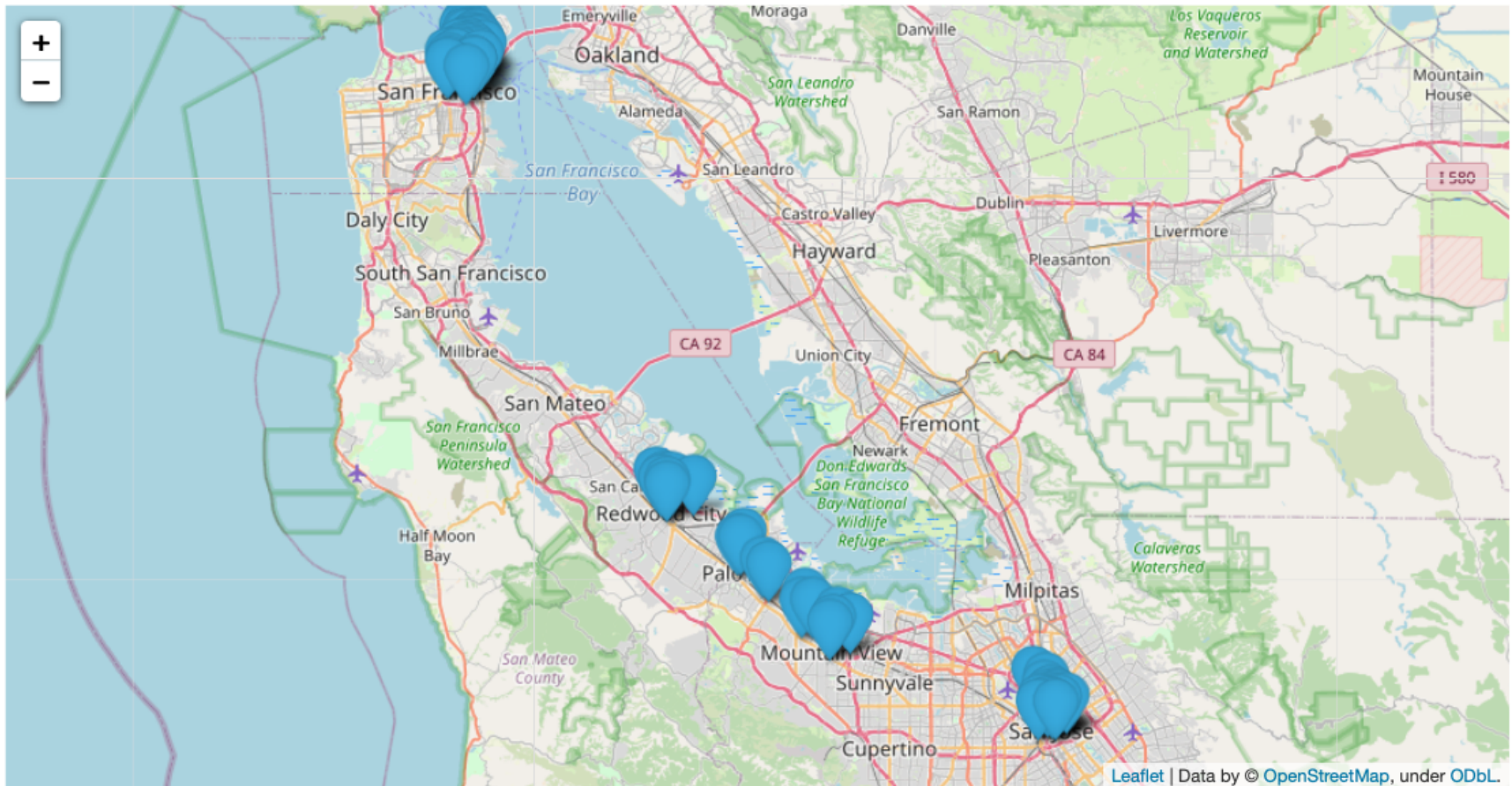
```
In [15]: # Show the San Francisco locations
         sf = stations.where('landmark', 'San Francisco')
         Circle.map_table(sf.select('lat', 'long', 'name'), color='green', area=100)
```

Bob Jacobsen, UC Berkeley
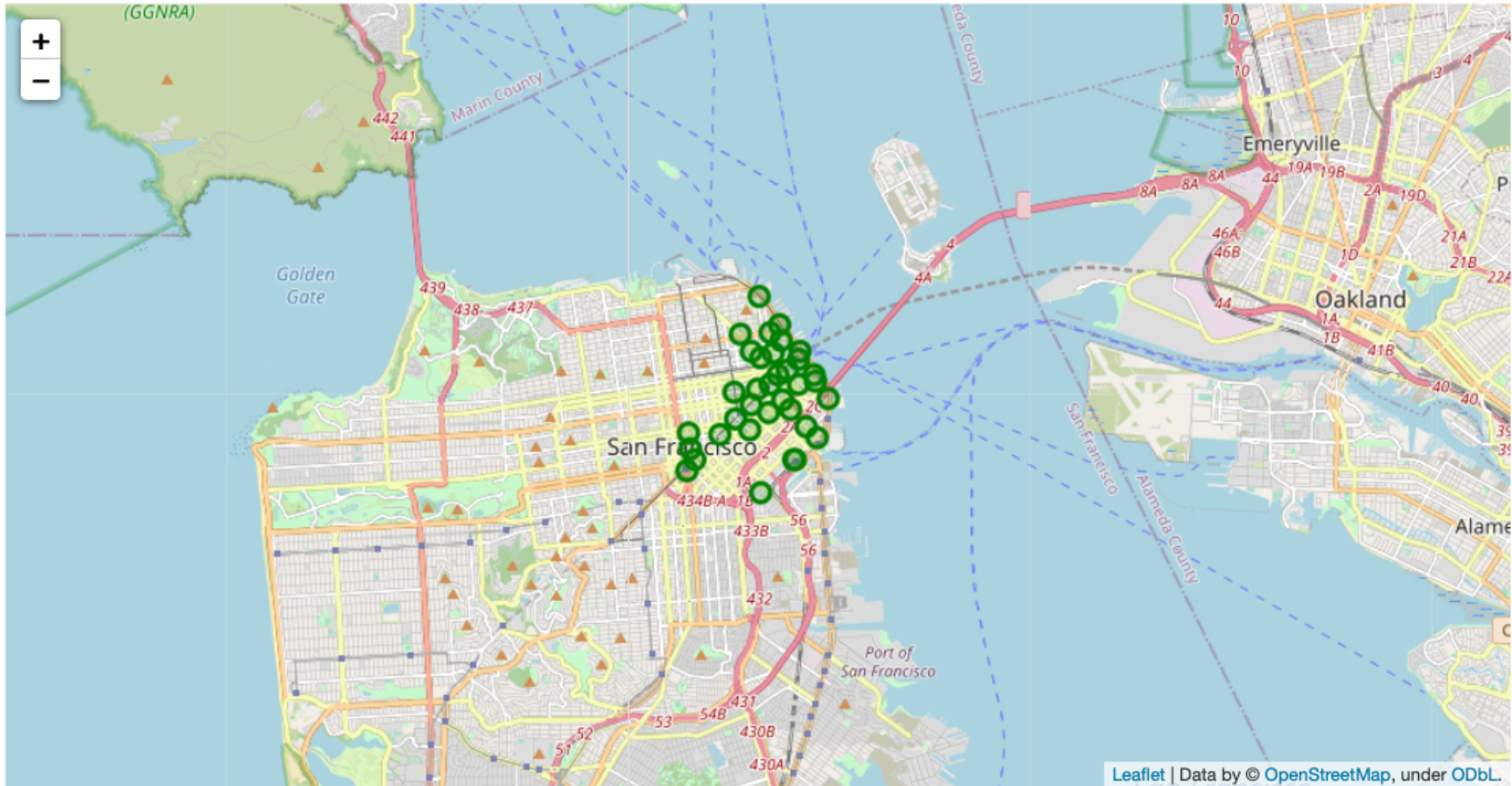
```
In [18]: # Calculate the number of trips starting at each station by joining the two data sets
         station_starts = stations.join('name', starts, 'Start Station')
         station_starts
```

Out[18]:

name	station_id	lat	long	dockcount	landmark	installation	count
2nd at Folsom	62	37.7853	-122.396	19	San Francisco	8/22/2013	7841
2nd at South Park	64	37.7823	-122.393	15	San Francisco	8/22/2013	9274
2nd at Townsend	61	37.7805	-122.39	27	San Francisco	8/22/2013	13674
5th at Howard	57	37.7818	-122.405	15	San Francisco	8/21/2013	7394
Adobe on Almaden	5	37.3314	-121.893	19	San Jose	8/5/2013	522
Arena Green / SAP Center	14	37.3327	-121.9	19	San Jose	8/5/2013	590
Beale at Market	56	37.7923	-122.397	19	San Francisco	8/20/2013	8135
Broadway St at Battery St	82	37.7985	-122.401	15	San Francisco	1/22/2014	7460
California Ave Caltrain Station	36	37.4291	-122.143	15	Palo Alto	8/14/2013	300
Castro Street and El Camino Real	32	37.386	-122.084	11	Mountain View	12/31/2013	1137

... (58 rows omitted)

Bob Jacobsen, UC Berkeley

In [19]:
```python
# Show how many trips start from each location?
Circle.map_table(station_starts.select('lat', 'long', 'name').with_columns(  # adding presentation options
    'color', 'blue',                                                          # show blue circles
    'area', station_starts.column('count') * 0.1                              # set circle size from number starts
))
```

Out[19]:

Bob Jacobsen, UC Berkeley
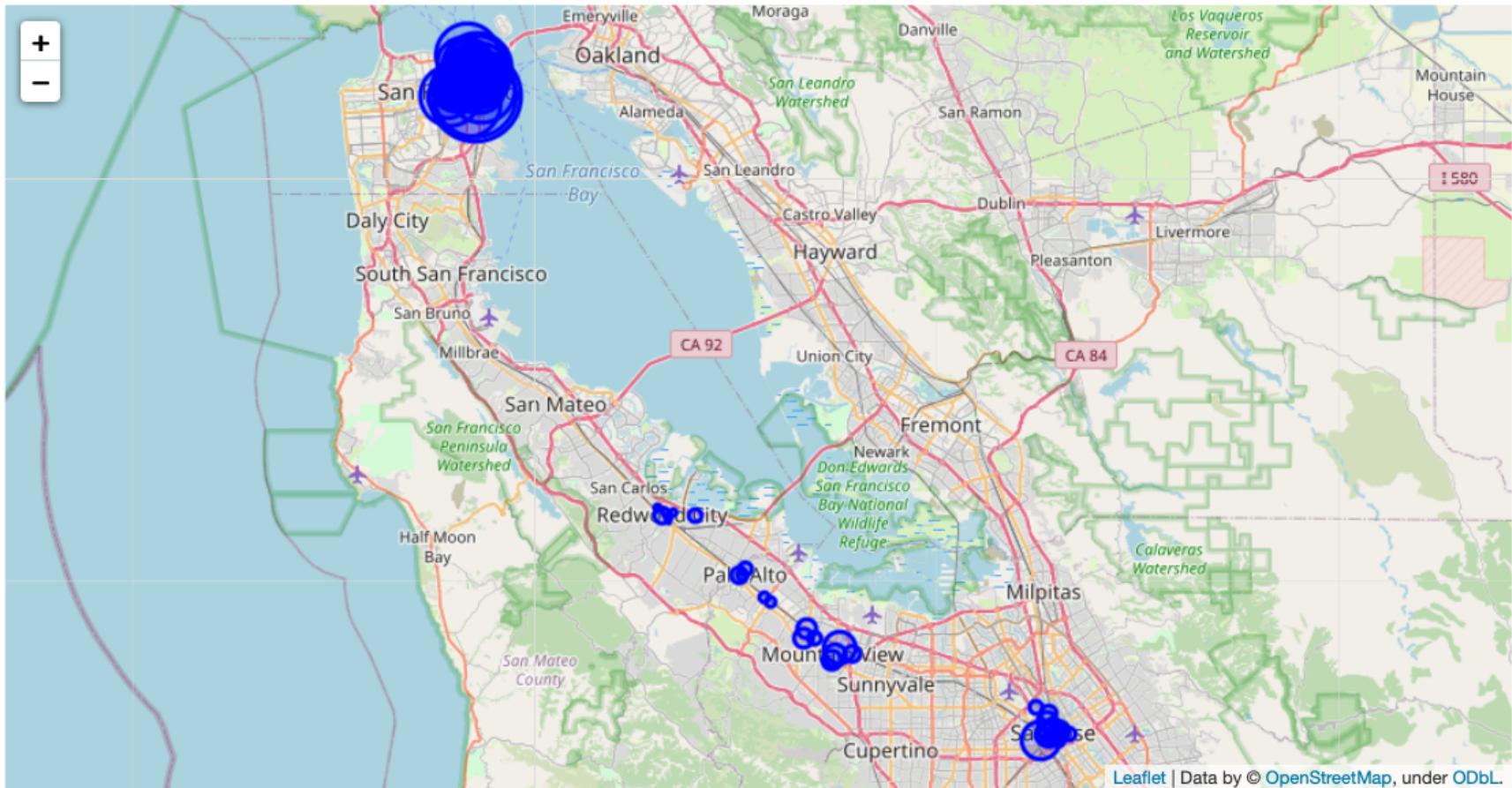
Outside the Box: Text Analysis

```
In [1]:  # Examine the book "Little Women" to see what we can learn from its text

         # usual imports
         from datascience import *
         import numpy as np
         import pandas as pd
         %matplotlib inline
         import matplotlib.pyplot as plots
         plots.style.use('fivethirtyeight')
         import warnings
         warnings.simplefilter(action="ignore", category=FutureWarning)

         from urllib.request import urlopen
         import re
         def read_url(url):
             return re.sub('\\s+', ' ', urlopen(url).read().decode())
```

```
In [2]:  # Read the book and split into separate chapters
         little_women_url = 'http://data8.org/materials-fa17/lec/little_women.txt'
         little_women_text = read_url(little_women_url)
         chapters = little_women_text.split('CHAPTER ')[1:]
```

```
In [3]:  # create a table with one chapter's text in each row
         Table().with_column('Text', chapters)
```

Out[3]:

Text
ONE PLAYING PILGRIMS "Christmas won't be Christmas witho ...
TWO A MERRY CHRISTMAS Jo was the first to wake in the gr ...
THREE THE LAURENCE BOY "Jo! Jo! Where are you?" cried Me ...
FOUR BURDENS "Oh, dear, how hard it does seem to take up ...
FIVE BEING NEIGHBORLY "What in the world are you going t ...
SIX BETH FINDS THE PALACE BEAUTIFUL The big house did pr ...
SEVEN AMY'S VALLEY OF HUMILIATION "That boy is a perfect ...
EIGHT JO MEETS APOLLYON "Girls, where are you going?" as ...
NINE MEG GOES TO VANITY FAIR "I do think it was the most ...
TEN THE P.C. AND P.O. As spring came on, a new set of am ...

... (37 rows omitted)

```
In [4]:  # Simple check:  Count the number of times "Christmas" appears in each chapter
         np.char.count(chapters, 'Christmas')

Out[4]:  array([8, 9, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 6, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                0])
```

```
In [5]:  # Count the number of times the characters' names appear in each chapter
         # and make a table with a column for each character
         references = Table().with_columns([
             "Jo",    np.char.count(chapters, "Jo"),
             "Meg",   np.char.count(chapters, "Meg"),
             "Amy",   np.char.count(chapters, "Amy"),
             "Beth",  np.char.count(chapters, "Beth"),
             "Laurie", np.char.count(chapters, "Laurie")
         ])
         references
```
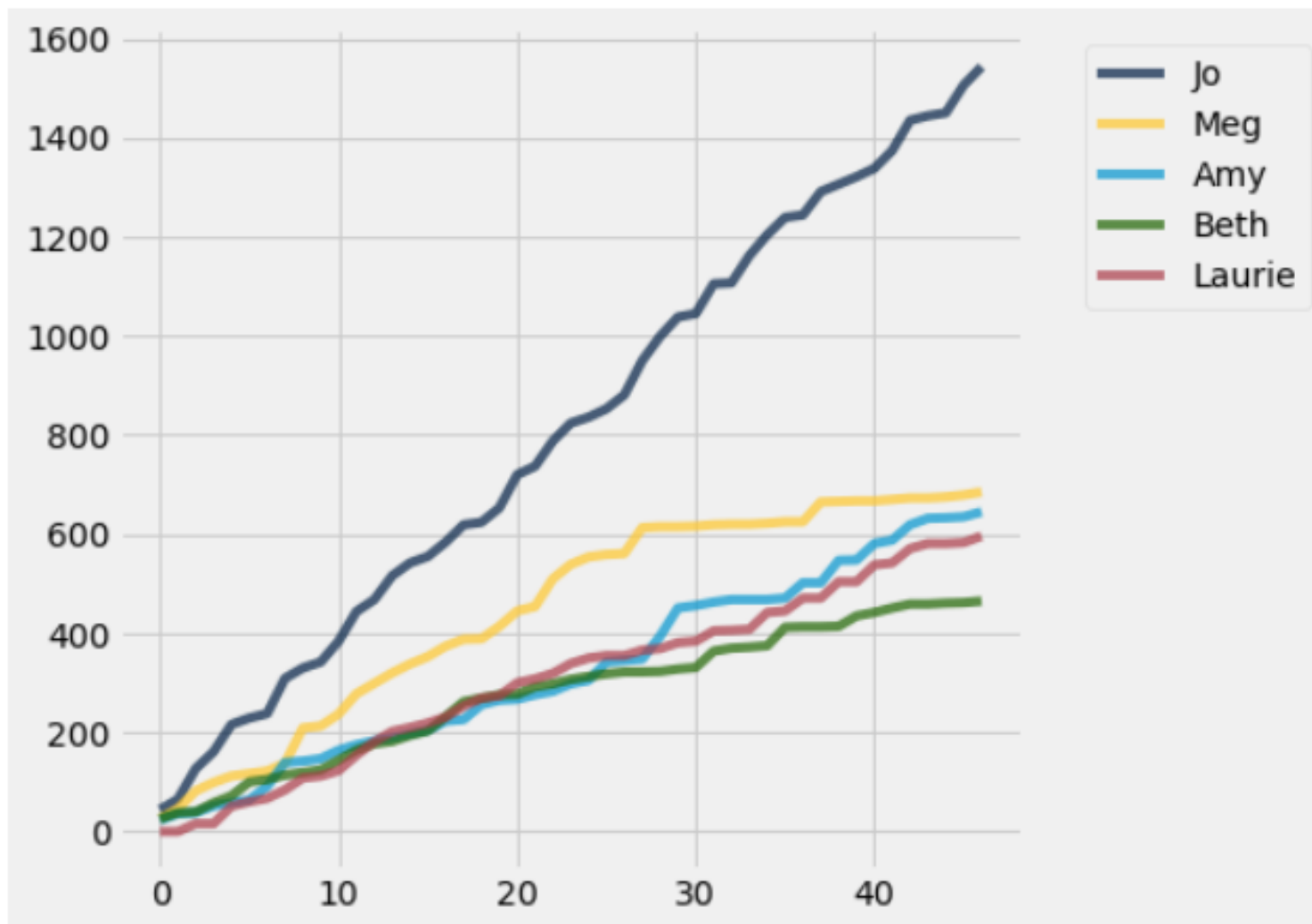
Out[5]:

Jo	Meg	Amy	Beth	Laurie
44	26	23	26	0
21	20	13	12	0
62	36	2	2	16
34	17	14	18	0
55	13	6	14	35
13	5	6	28	9
9	5	27	5	7
71	16	48	9	17
21	71	3	5	24
12	4	5	5	4

... (37 rows omitted)

In [6]:
```python
# plot appearances by chapter, one curve per character (column)
references.plot()
```

Bob Jacobsen, UC Berkeley
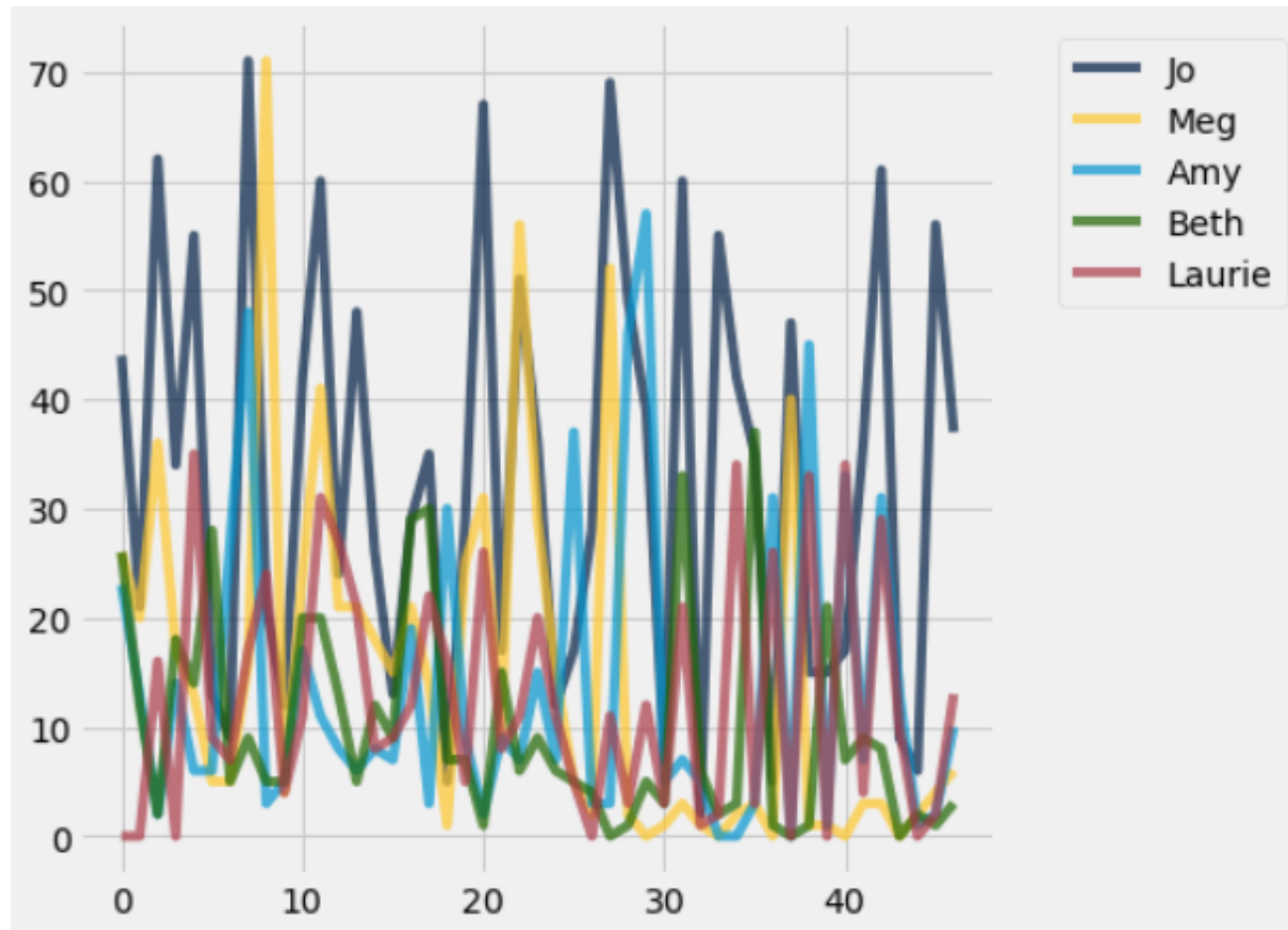
In [7]:
```python
# the plot-by-chapter is hard to interpret.  Plot cumulative sums:
references.cumsum().plot()
```


In [8]:
```python
# How would you see who's mentioned most often in each chapter?
```

Bob Jacobsen, UC Berkeley

Notebooks as persuasive objects

"When you two have finished arguing your opinions, I actually have data!"

Bob Jacobsen, UC Berkeley

Notebooks as persuasive objects

You've seen some of this already:

Plots and tables to show data

Links to document sources and background information

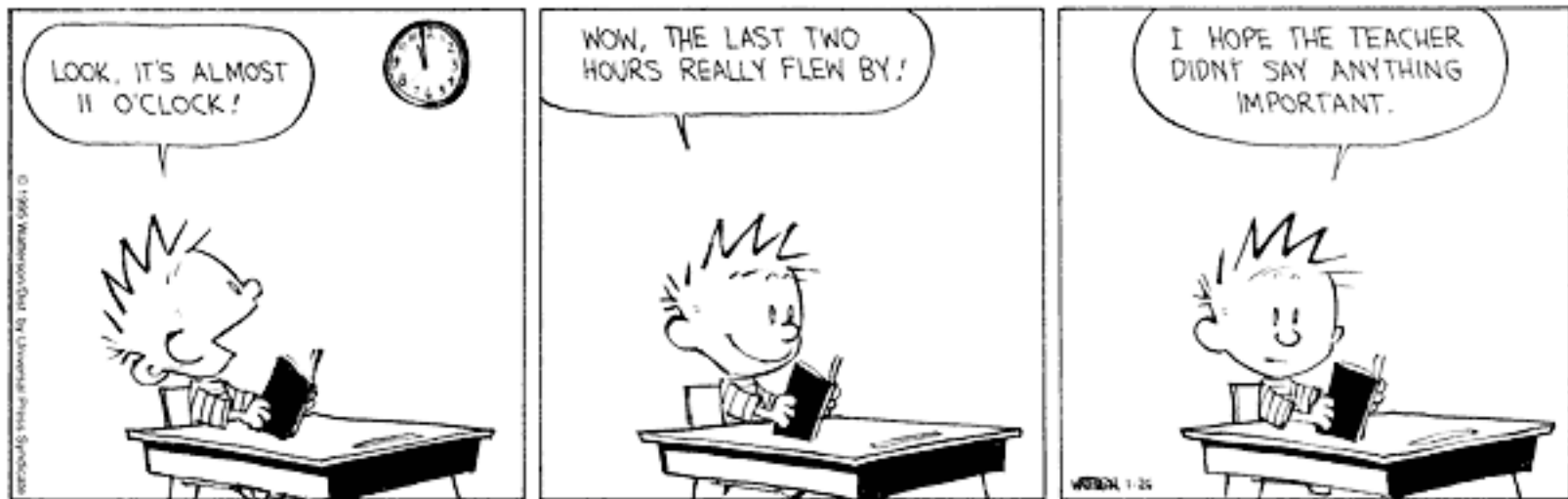Ability to rapidly respond to "what if" questions

Markdown for pretty titles and text

Bob Jacobsen, UC Berkeley

Notebooks as persuasive objects

You've seen some of this already:

Bob Jacobsen, UC Berkeley

Bob Jacobsen, UC Berkeley

Bob Jacobsen, UC Berkeley
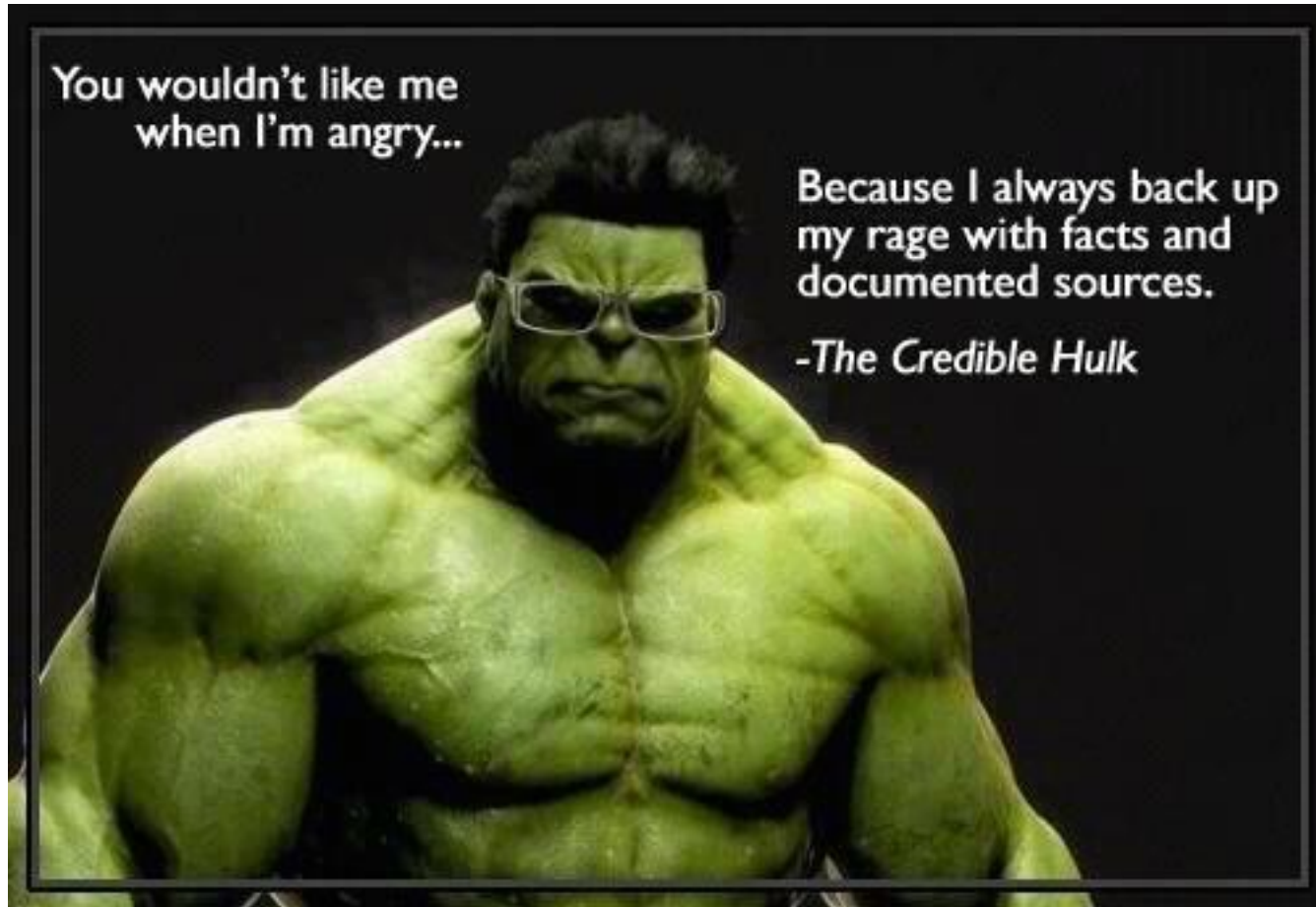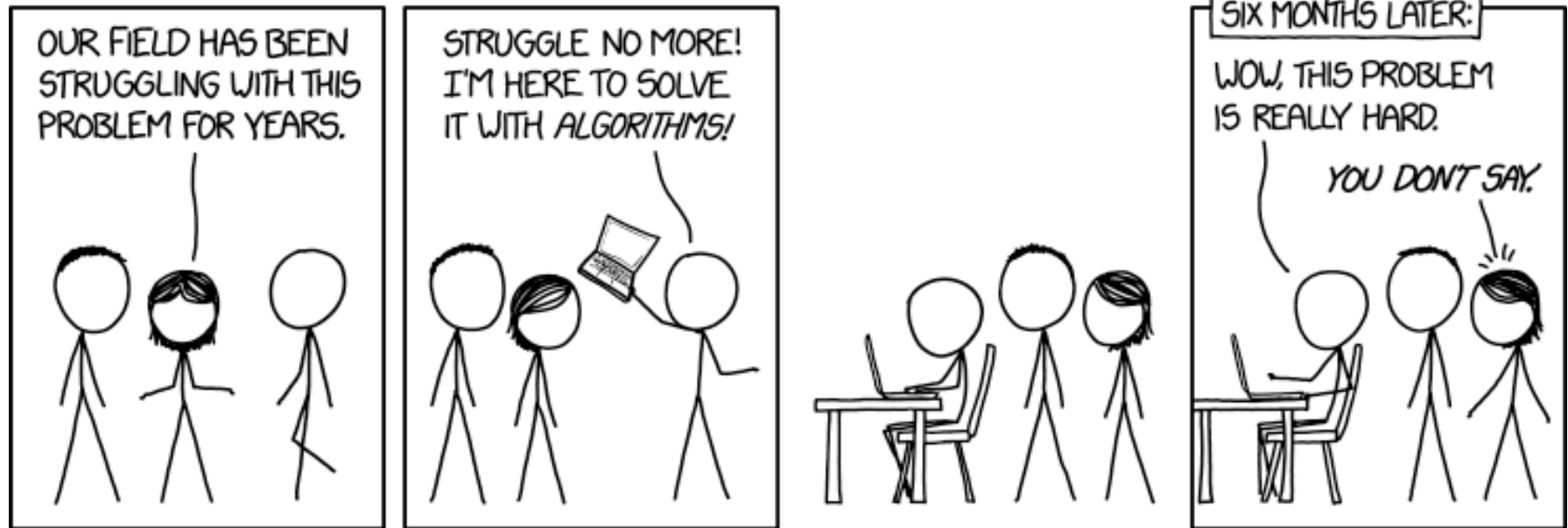
<u>Exercises on Wednesday</u>

Intro - these notebooks & the SWAN service

Simple Applications

Project(s)!

Instructions to get started on Indigo (Data Science E1)

https://indico.cern.ch/event/1125271/contributions/4975773/

If you get stuck, ask for help or do an internet search

Learn about each topic, spend more time on ones that interest you.

Don't try to do every bit of every notebook; pick interesting ones.

Speed is not the issue: no reward for first done or most complete coverage

　　Not even keeping track

<u>Think</u> about what you're doing: Learn to use these tools!

Bob Jacobsen, UC Berkeley