Data Science 1

STAT/CS 287

James Bagrow, UVM Dept of Math and Statistics

LECTURE 01

Today’s Schedule

Course logistics

Intro + Motivation

Intro/Review of programming

Homework

1. Register and install Anaconda Python on your machine (see Python Setup slides)
→ Complete **HW01**
→ HW01 includes additional setup reading

2. Work through **reading**—there's a lot but the more prepared you are now, the faster you can work later

Quiz 01 will assess your preparation following this reading

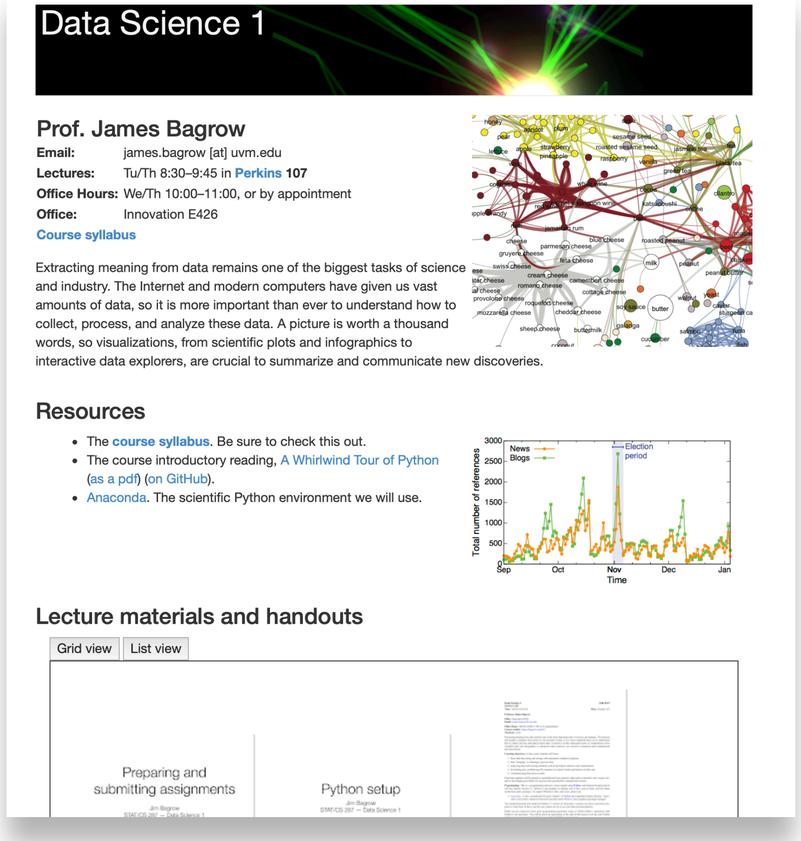
(will discuss schedule of assignments and quizzes shortly)

Course logistics

Website: bagrow.com/ds1/

Syllabus

Lecture notes/slides



Assignments on **Blackboard**

- If you need to know when something is due, when is the quiz, etc., Blackboard has the truth!
- Check Blackboard regularly—*you're responsible for keeping track of assignment postings, updates to assignments, due dates, etc.*

Course logistics

COVID-19

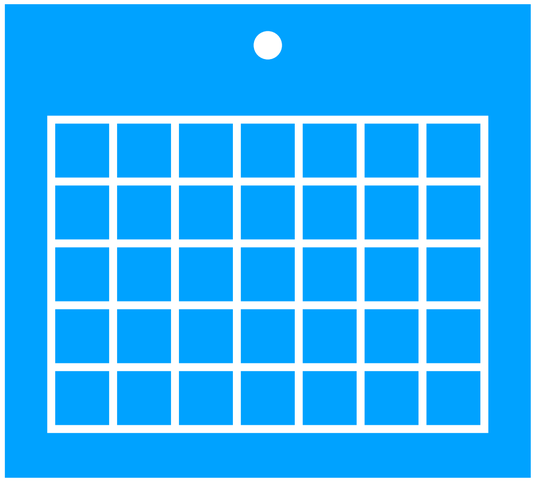
- Flexible and accommodating of attendance this semester → but keep on top of work!

- Aim to record lectures (from laptop), post within 24 hours
- Future is uncertain, but right now nothing is so urgent that we need *real-time* remote

Course logistics

Let’s go over the syllabus right now!

Tentative schedule



Project details coming soon!

Week	Assigned	Due	Quiz
1	HW 01		
2	HW 02	HW 01	Q 01
3	HW 03	HW 02	
4	HW 04	HW 03	Q 02
5	Project 1	HW 04	
6			Q 03
7	HW 05	Project 1	
8	HW 06	HW 05	Q 04
9	HW 07	HW 06	
10	Project 2	HW 07	
11			Q 05
12	HW 08	Project 2	
13	*Thanksgiving Recess*		
14	HW 09	HW 08	Q 06
15		HW 09	
Finals		Final project	

Lectures consist of

Slides

Data Science 1

STAT/CS 287
Prof. James Bagrow, UVM Dept of Math and Statistics

LECTURE 01

Notebooks

DS1 Lecture 01 supplement

Jim Bagrow

This notebook acts as an additional supplement on programming in Python. You are expected to learn this material during the first week of class, and the first quiz of the semester will assess this.

This document complements the course's **"Whirlwind Tour" reading**. You are also expected to complete this reading assignment. I recommend you review this supplement after reading the whirlwind tour.

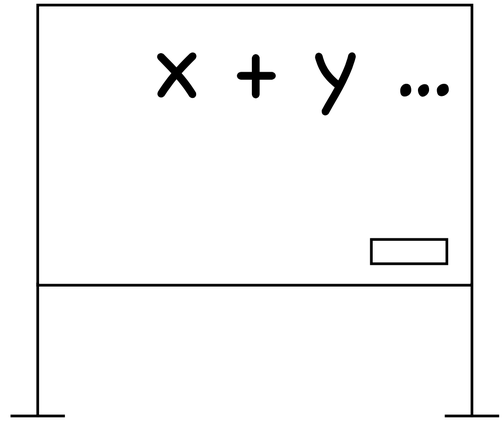
About this document

- This is a **Jupyter Notebook**. I'll be using these for code-heavy lectures and lecture supplements. Notebooks contain IPython-style input and output cells in a document, interleaved with narrative (non-code) text.
- Your **projects and assignments** will be submitted as Python scripts (`.py` files) with associated writeups (`.docx` files).

Contents:

- A bit more on data structures and control flow (see also the Whirlwind Tour)
  - tuples vs. lists
    - multiassignment
  - sorting
  - zipping
- Working with **files** !
- Exploring the Python standard library (builtin modules and packages)
- A tour of useful third-party packages [time permitting]

Board



Lectures consist of

Slides

Data Science 1

STAT/CS 287
Prof. James Bagrow, UVM Dept of Math and Statistics

LECTURE 01

Notebooks

DS1 Lecture 01 supplement

Jim Bagrow

This notebook acts as an additional supplement on programming in Python. You are expected to learn this material during the first week of class, and the first quiz of the semester will assess this.

This document complements the course's **"Whirlwind Tour" reading**. You are also expected to complete this reading assignment. I recommend you review this supplement after reading the whirlwind tour.

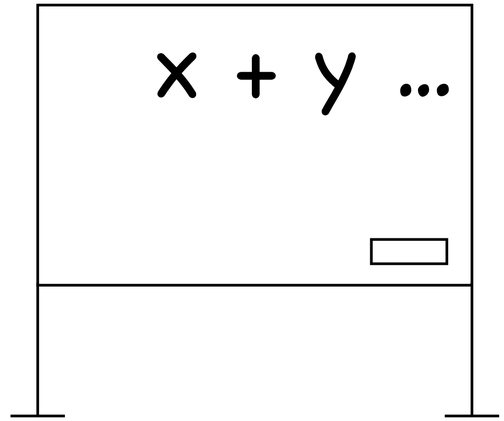
About this document

- This is a **Jupyter Notebook**. I'll be using these for code-heavy lectures and lecture supplements. Notebooks contain IPython-style input and output cells in a document, interleaved with narrative (non-code) text.
- Your **projects and assignments** will be submitted as Python scripts (`.py` files) with associated writeups (`.docx` files).

Contents:

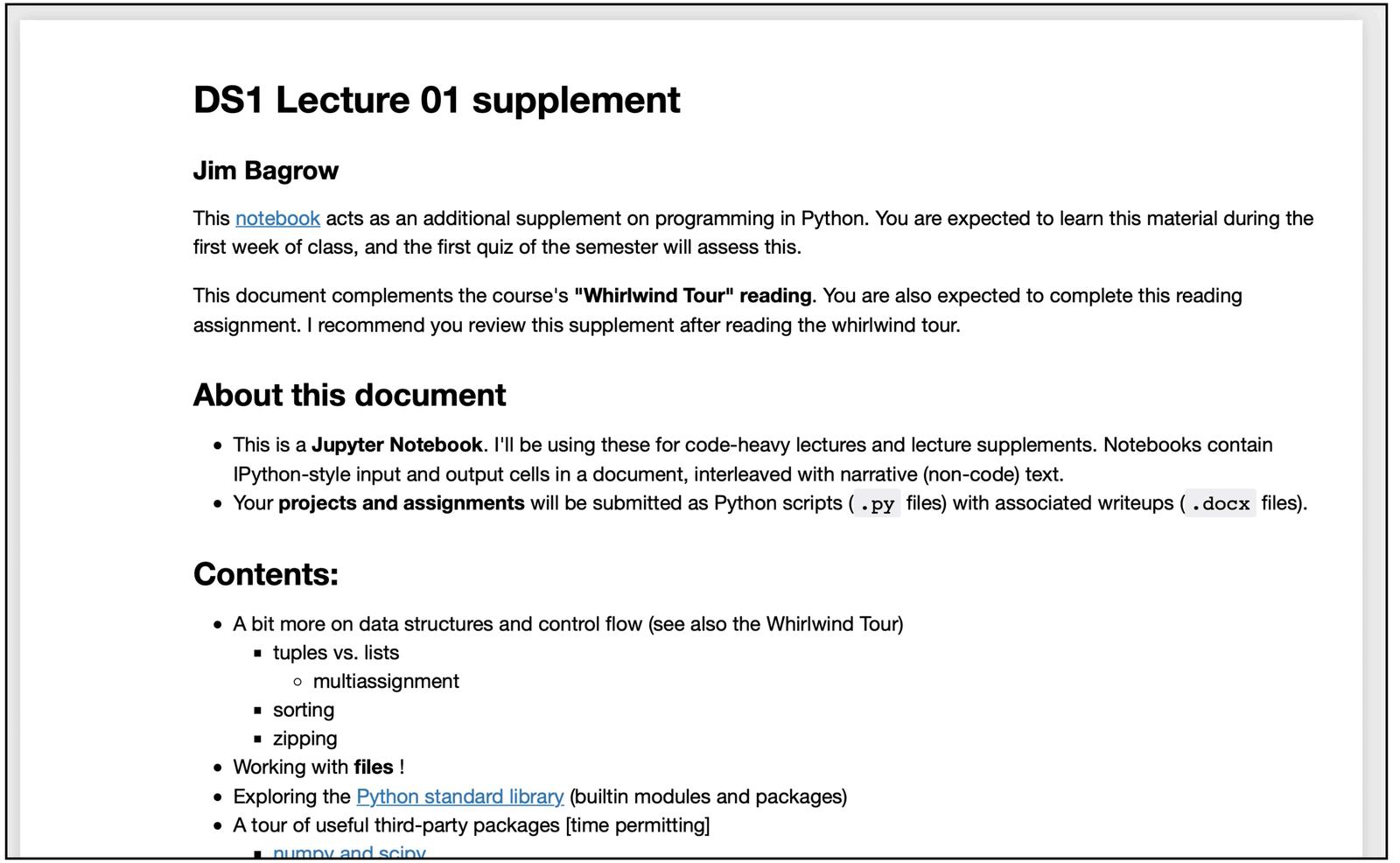
- A bit more on data structures and control flow (see also the Whirlwind Tour)
  - tuples vs. lists
    - multiassignment
  - sorting
  - zipping
- Working with **files** !
- Exploring the Python standard library (builtin modules and packages)
- A tour of useful third-party packages [time permitting]

Board



Lectures consist of

Notebooks



DS1 Lecture 01 supplement

Jim Bagrow

This notebook acts as an additional supplement on programming in Python. You are expected to learn this material during the first week of class, and the first quiz of the semester will assess this.

This document complements the course's **"Whirlwind Tour" reading**. You are also expected to complete this reading assignment. I recommend you review this supplement after reading the whirlwind tour.

About this document

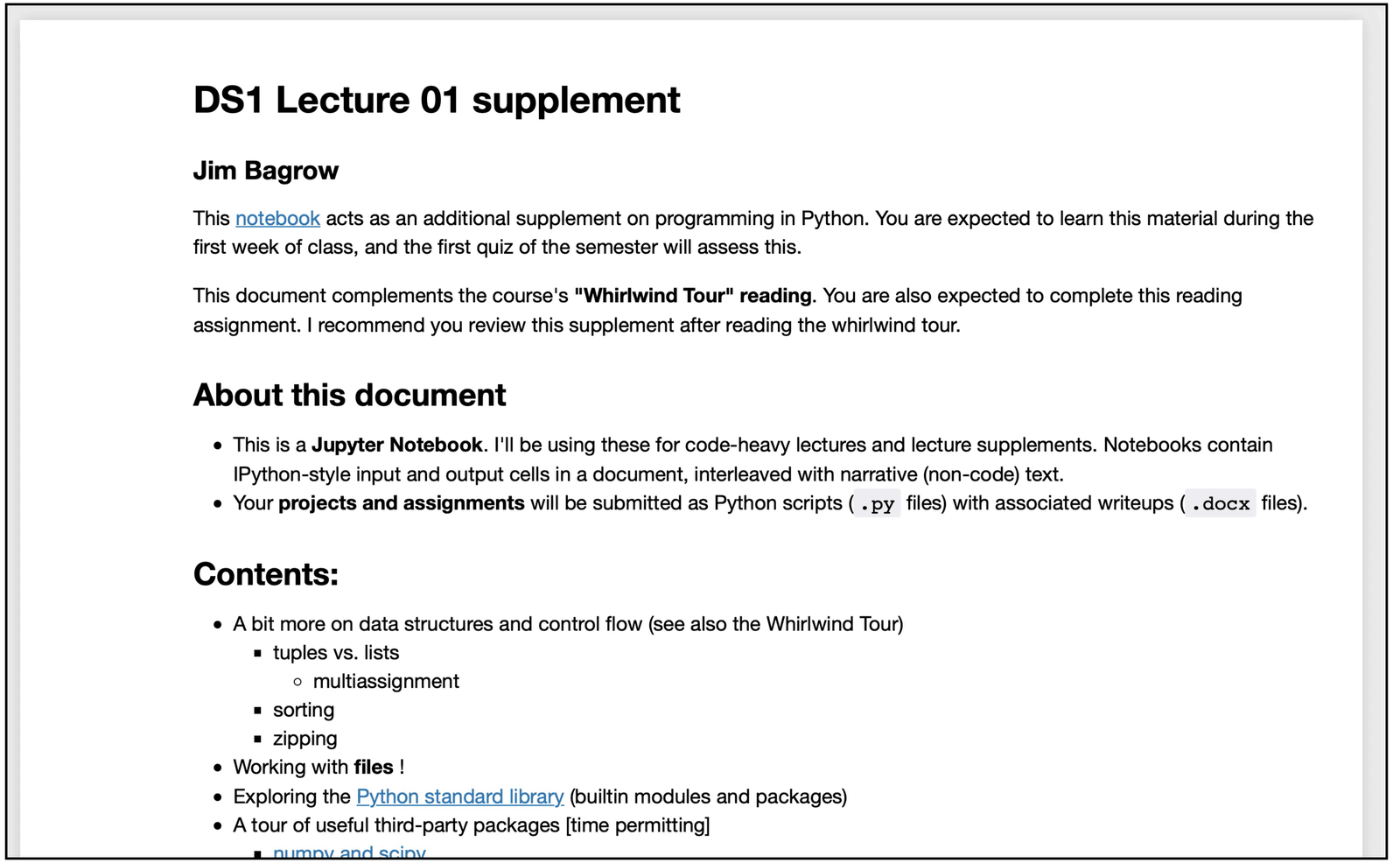
- This is a **Jupyter Notebook**. I'll be using these for code-heavy lectures and lecture supplements. Notebooks contain IPython-style input and output cells in a document, interleaved with narrative (non-code) text.
- Your **projects and assignments** will be submitted as Python scripts (`.py` files) with associated writeups (`.docx` files).

Contents:

- A bit more on data structures and control flow (see also the Whirlwind Tour)
  - tuples vs. lists
    - multiassignment
  - sorting
  - zipping
- Working with **files** !
- Exploring the Python standard library (builtin modules and packages)
- A tour of useful third-party packages [time permitting]
  - numpy and scipy

Lectures consist of

Notebooks



DS1 Lecture 01 supplement

Jim Bagrow

This notebook acts as an additional supplement on programming in Python. You are expected to learn this material during the first week of class, and the first quiz of the semester will assess this.

This document complements the course's **"Whirlwind Tour" reading**. You are also expected to complete this reading assignment. I recommend you review this supplement after reading the whirlwind tour.

About this document

- This is a **Jupyter Notebook**. I'll be using these for code-heavy lectures and lecture supplements. Notebooks contain IPython-style input and output cells in a document, interleaved with narrative (non-code) text.
- Your **projects and assignments** will be submitted as Python scripts (`.py` files) with associated writeups (`.docx` files).

Contents:

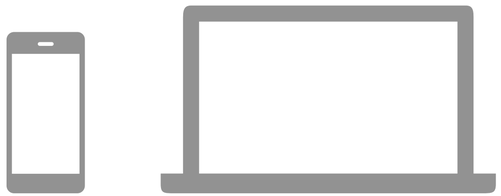
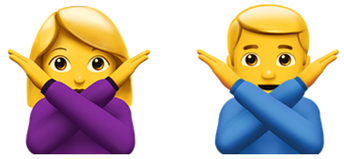
- A bit more on data structures and control flow (see also the Whirlwind Tour)
  - tuples vs. lists
    - multiassignment
  - sorting
  - zipping
- Working with **files** !
- Exploring the Python standard library (builtin modules and packages)
- A tour of useful third-party packages [time permitting]
  - numpy and scipy

How to succeed in this class

- Be active
- Think critically
- Start early on assignments
- Focus on initial review materials
- Organize and summarize information yourself

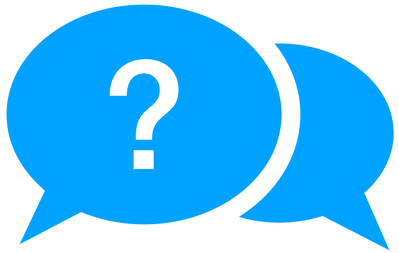
How to succeed in this class

- Be active
- Think critically
- Start early on assignments
- Focus on initial review materials
- Organize and summarize information yourself

Strongly advise you to avoid electronic devices during class

Course logistics



Questions?

Today's Schedule

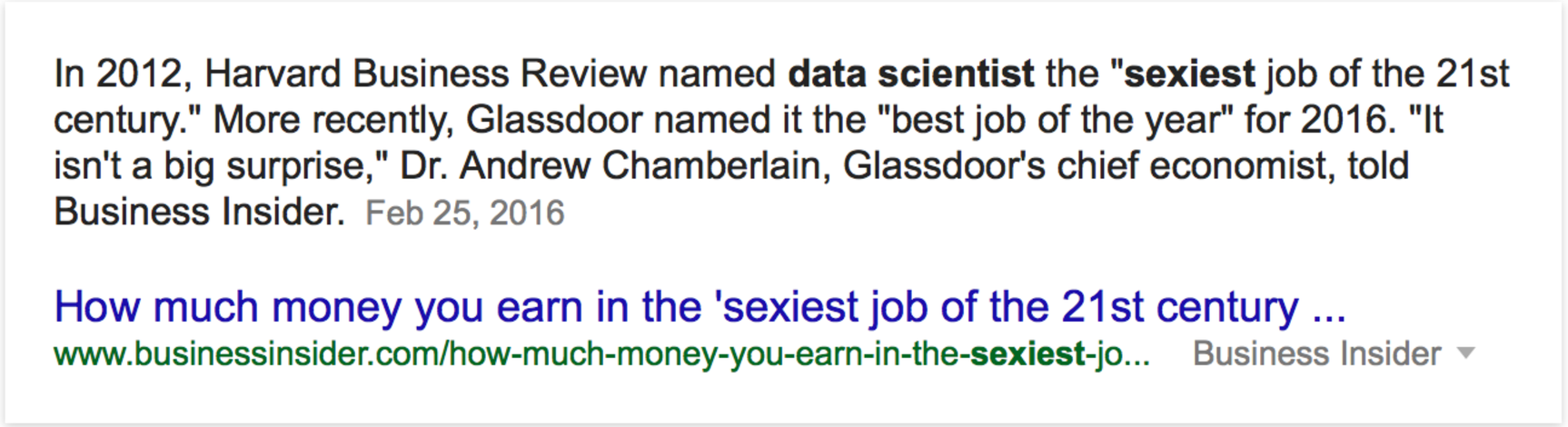
- ✓ Course logistics
- Intro + Motivation
- Intro/Review of programming

Intro + Motivation

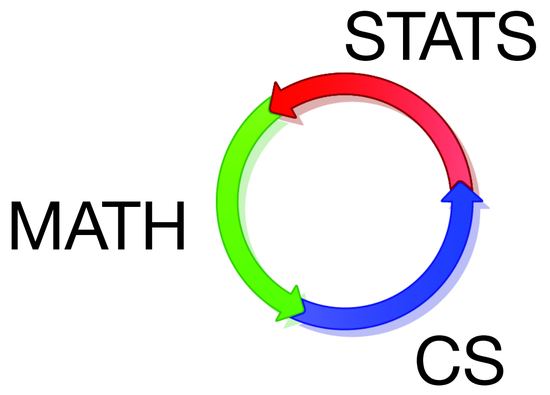
What is data science?

“Data science is the study of the generalizable extraction of knowledge from data”

http://en.wikipedia.org/wiki/Data_science

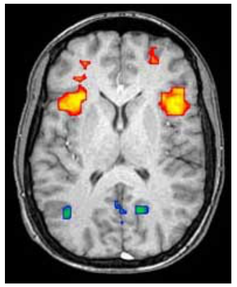


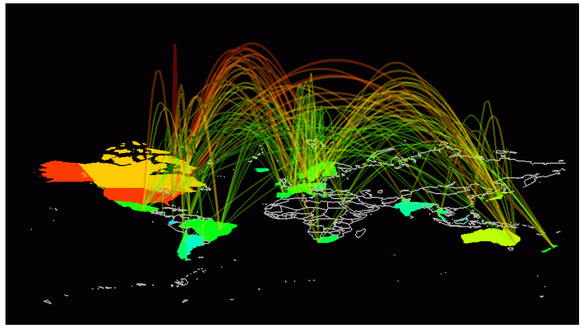
What is data science?



mixed with **domain knowledge**:

- social networks/sociology
- biology or drug discovery
- neuroscience
- business
- etc…



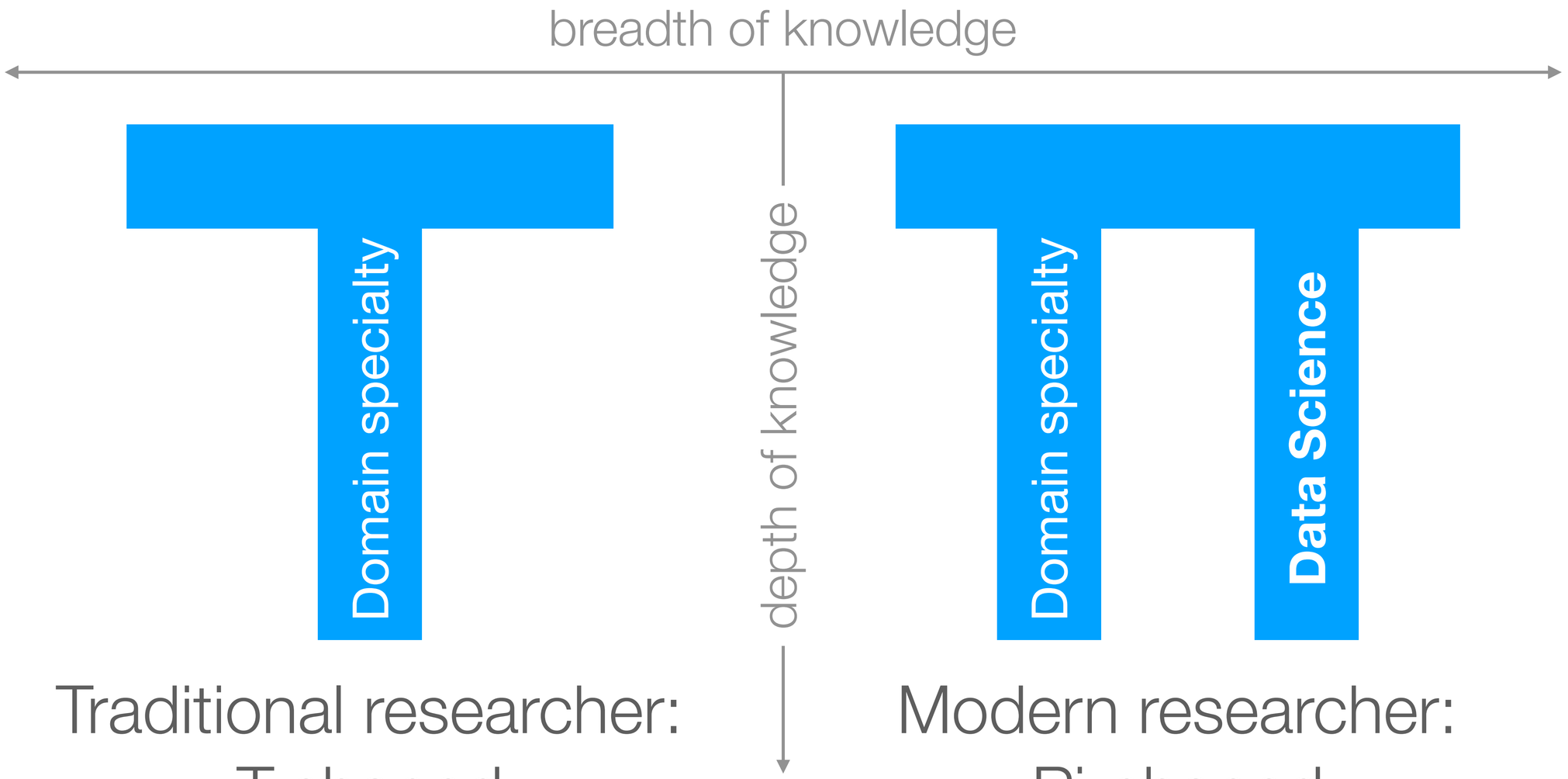


What is data science?

A **meta-field** field encompassing:

statistics
machine learning
data mining
knowledge discovery
database engineering
information retrieval
visualization
⋮

What is data science?



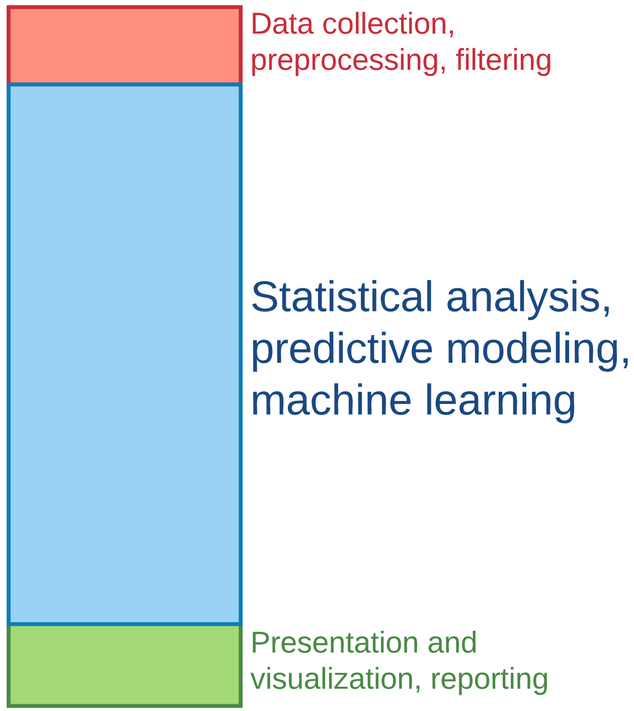
Traditional researcher:
T-shaped

Modern researcher:
Pi-shaped

* Alexander Szalay

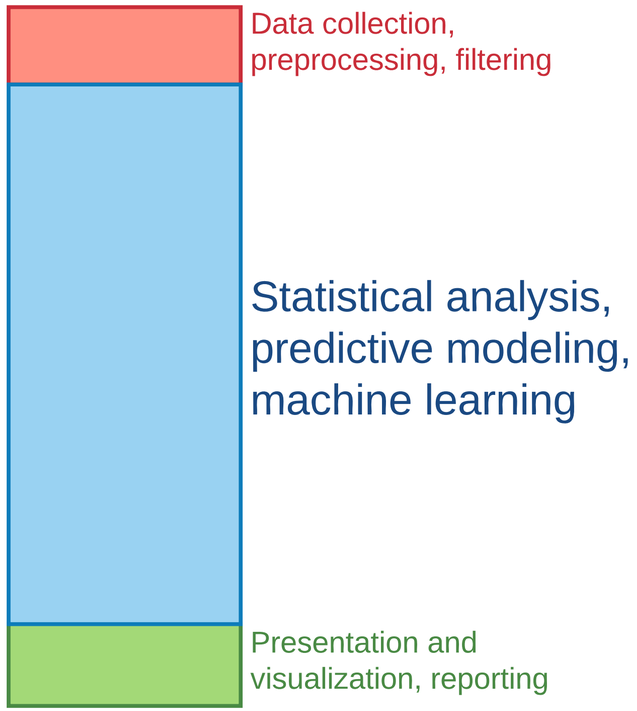
What is data science?

What research and coursework make data science **look like**

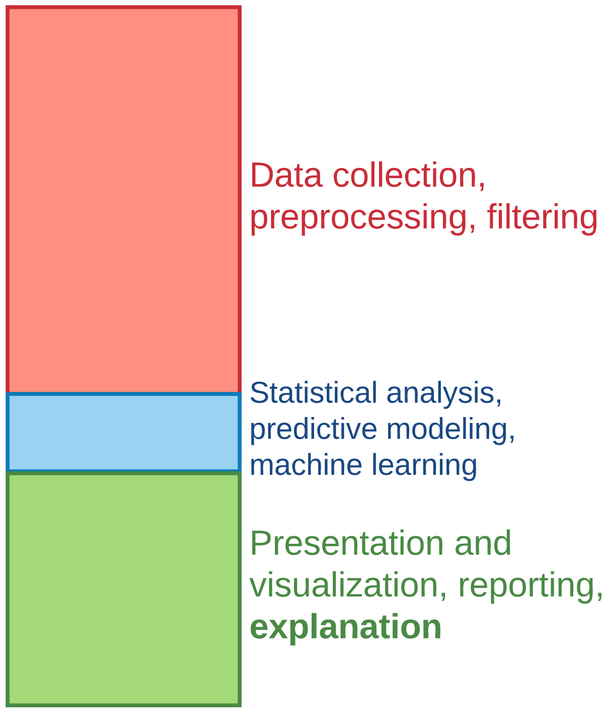


What is data science?

What research and coursework make data science **look like**



What data science **actually is**



Course topics include

- "good" practices for scientific computing
- practical data exploration and analysis
- clean and process data (data dexterity)
- web access to data sets, data "APIs"
- Natural language processing (NLP) and machine learning
- Classic regression and inference methods
- working with missing and malformed data
- Bayesian statistics and computation
- visualization principles and design
- Interpreting and communicating results!

A data science question

A data science question

Twitter is famous for using very short messages called tweets. Tweets were originally limited to **140 characters**

Short messages are easy to read, but not always easy to write

A data science question

Twitter is famous for using very short messages called tweets. Tweets were originally limited to **140 characters**

Short messages are easy to read, but not always easy to write

Question: Is 140 characters too short?

A data science question

Twitter went many years with 140 characters. Why 140?

A data science question

Twitter went many years with 140 characters. Why 140?

"Twittr" in 2006:

A data science question

Twitter went many years with 140 characters. Why 140?

"Twittr" in 2006:

"If you have a cell and can txt…"

A data science question

Twitter went many years with 140 characters. Why 140?

Twitter was originally focused on text messaging (SMS)

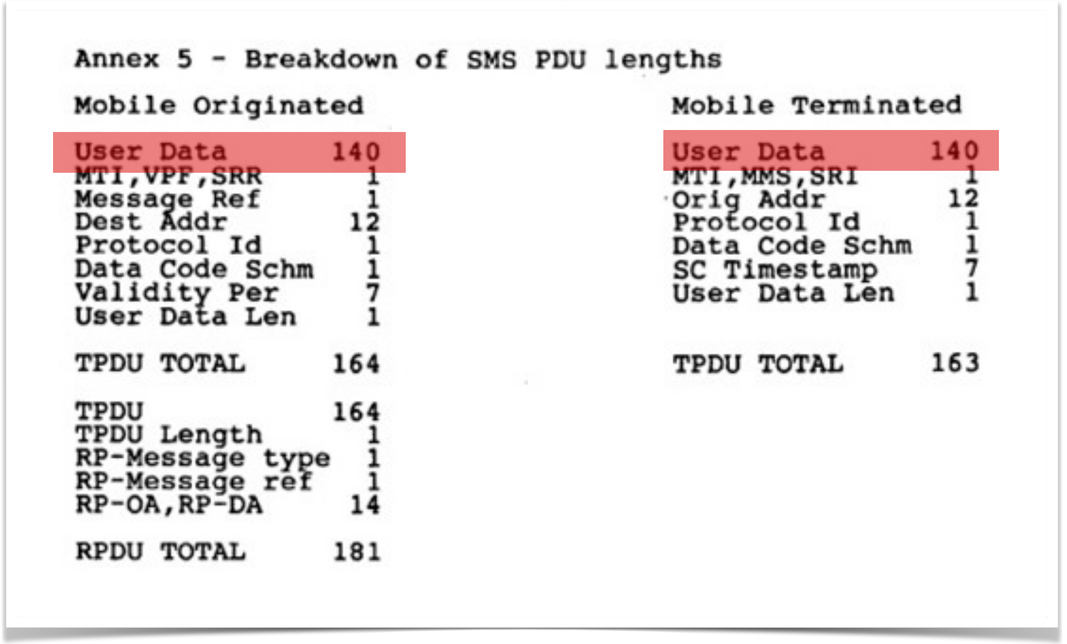
The SMS communication protocol (over 30 years old!) specifies the "payload" size for each message:

A data science question

Twitter went many years with 140 characters. Why 140?

Twitter was originally focused on text messaging (SMS)

The SMS communication protocol (over 30 years old!) specifies the "payload" size for each message:



Annex 5 - Breakdown of SMS PDU lengths

Mobile Originated		Mobile Terminated	
User Data	140	User Data	140
MTI,VPF,SRR	1	MTI,MMS,SRI	1
Message Ref	1	Orig Addr	12
Dest Addr	12	Protocol Id	1
Protocol Id	1	Data Code Schm	1
Data Code Schm	1	SC Timestamp	7
Validity Per	7	User Data Len	1
User Data Len	1		
TPDU TOTAL	164	TPDU TOTAL	163
TPDU	164		
TPDU Length	1		
RP-Message type	1		
RP-Message ref	1		
RP-OA,RP-DA	14		
RPDU TOTAL	181		

Source: GSM4 126/91

A data science question

Question: Is 140 characters too short?

A data science question

Question: Is 140 characters too short?

Idea: We can look at people's tweets, how long are tweets?

A data science question

Question: Is 140 characters too short?

Idea: We can look at people's tweets, how long are tweets?

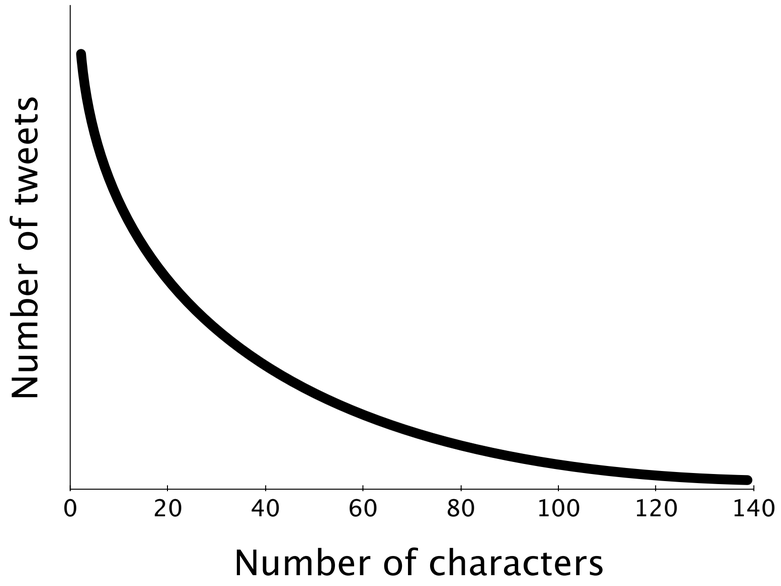
Hypothesis 1: most tweets are short,
almost none reach 140
→ Plenty of room

A data science question

Question: Is 140 characters too short?

Idea: We can look at people's tweets, how long are tweets?

Hypothesis 1: most tweets are short, almost none reach 140
→ Plenty of room

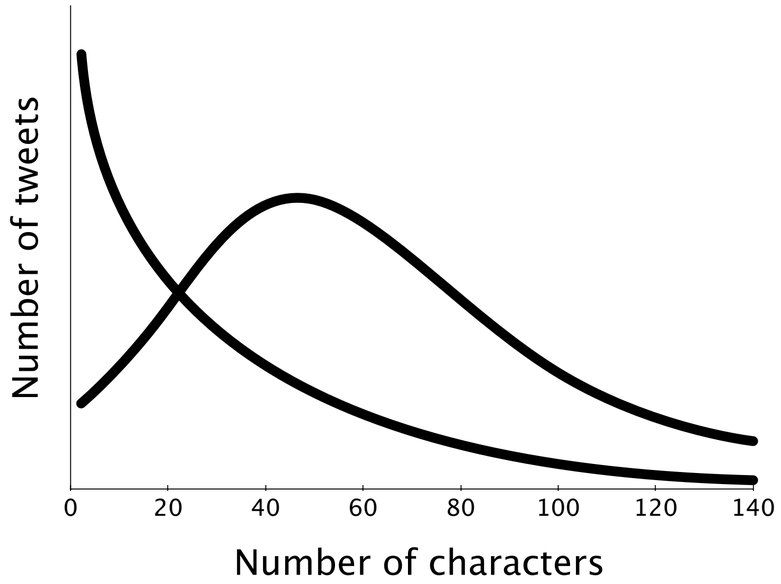


A data science question

Question: Is 140 characters too short?

Idea: We can look at people's tweets, how long are tweets?

Hypothesis 1: most tweets are short, almost none reach 140
→ Plenty of room



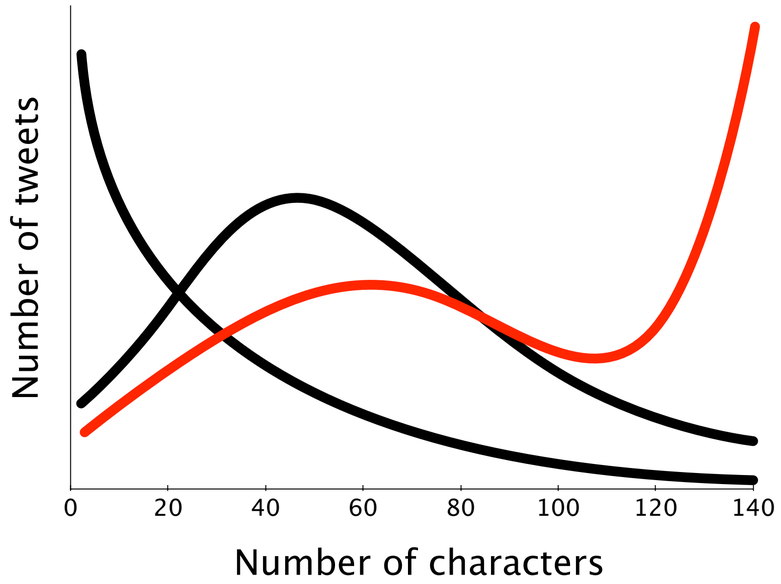
A data science question

Question: Is 140 characters too short?

Idea: We can look at people's tweets, how long are tweets?

Hypothesis 1: most tweets are short, almost none reach 140
→ Plenty of room

Hypothesis 2: many tweets hitting the 140 limit
→ People are *cramming* tweets



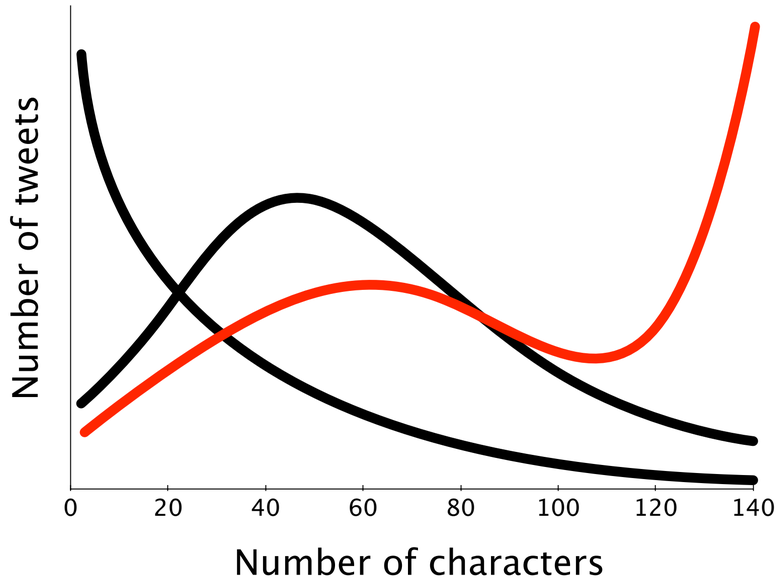
A data science question

Question: Is 140 characters too short?

Idea: We can look at people's tweets, how long are tweets?

Hypothesis 1: most tweets are short, almost none reach 140
→ Plenty of room

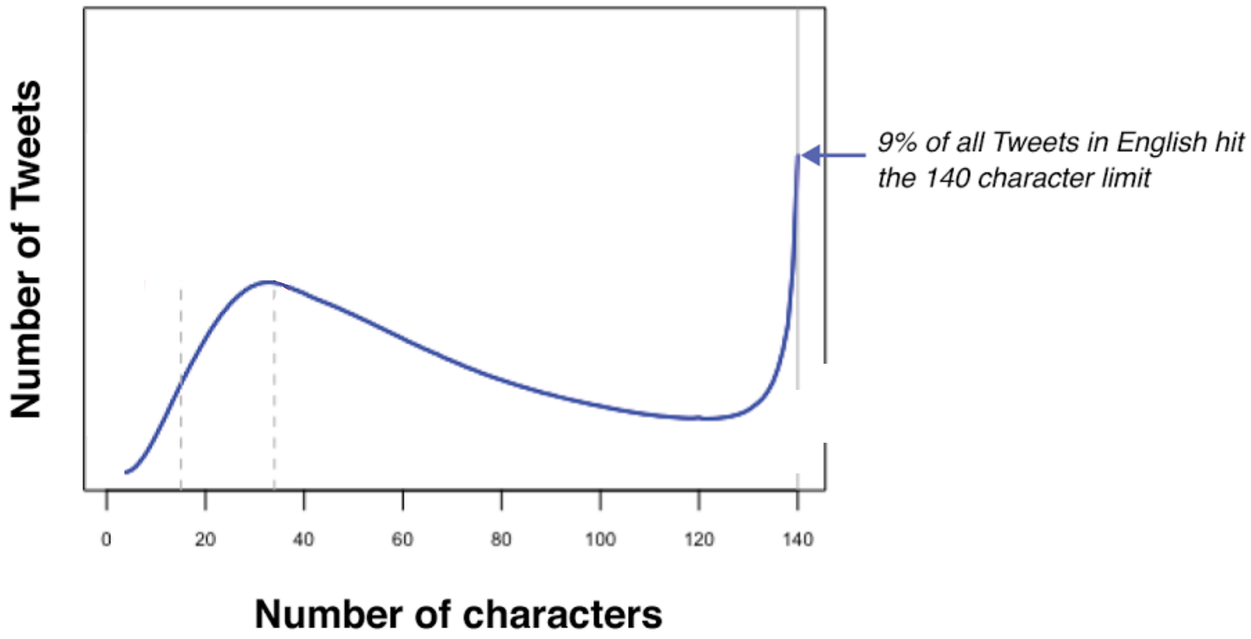
Hypothesis 2: many tweets hitting the 140 limit
→ People are *cramming* tweets



What's it really look like?

A data science question

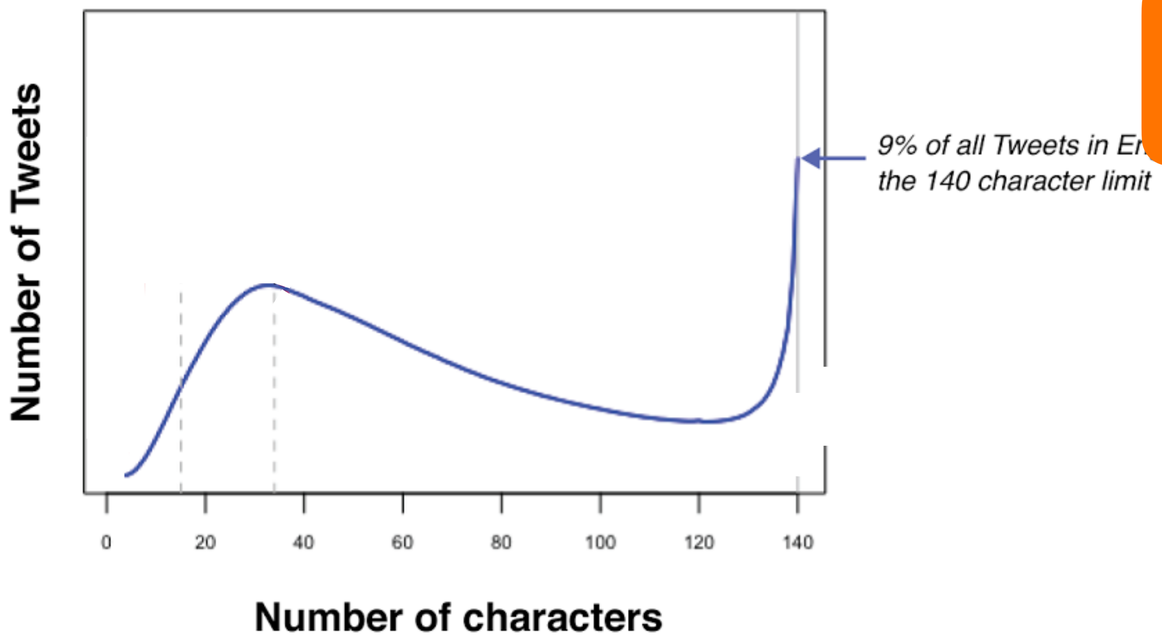
Question: Is 140 characters too short?



https://blog.twitter.com/en_us/topics/product/2017/Giving-you-more-characters-to-express-yourself.html

A data science question

Question: Is 140 characters too short?

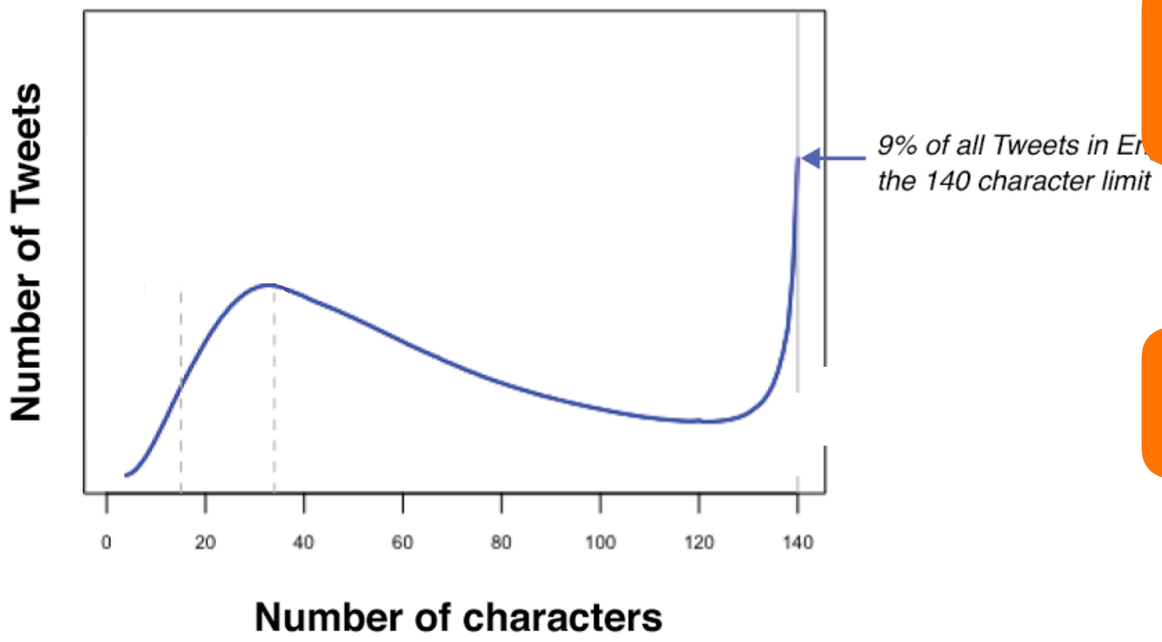


Answer: Yes, probably too short

https://blog.twitter.com/en_us/topics/product/2017/Giving-you-more-characters-to-express-yourself.html

A data science question

Question: Is 140 characters too short?



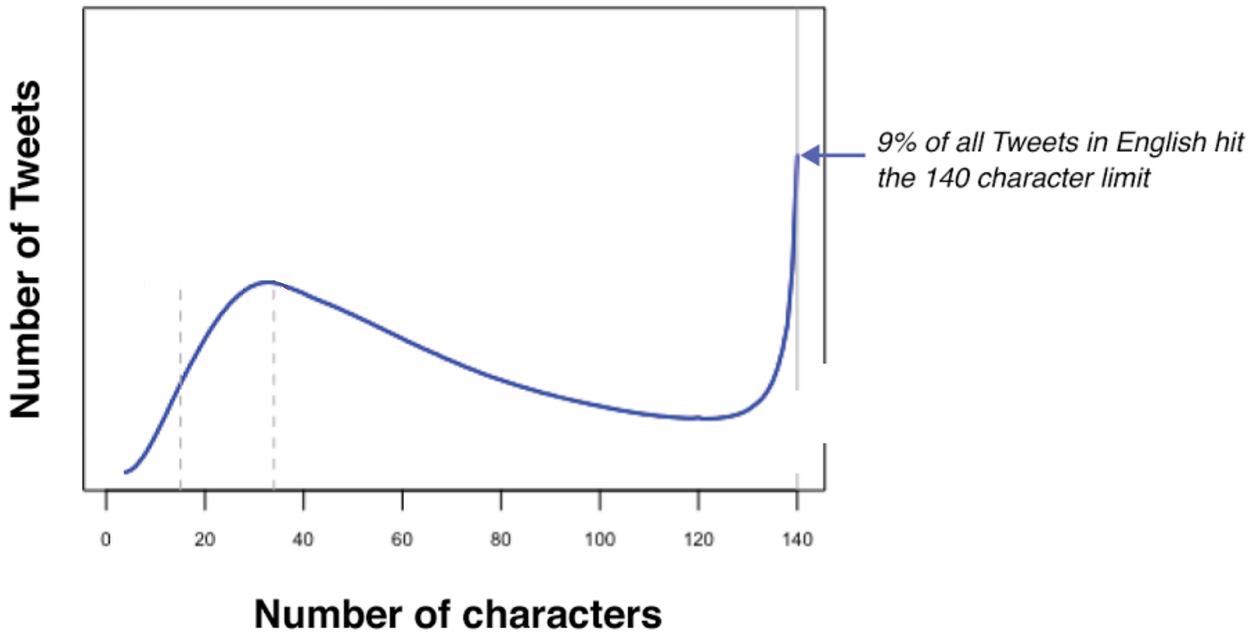
Answer: Yes, probably too short

New question: if the limit is raised, will people just cram into that new length? What should the limit be?

https://blog.twitter.com/en_us/topics/product/2017/Giving-you-more-characters-to-express-yourself.html

A data science question

New question: if the limit is raised, will people just cram into that new length? What should the limit be?

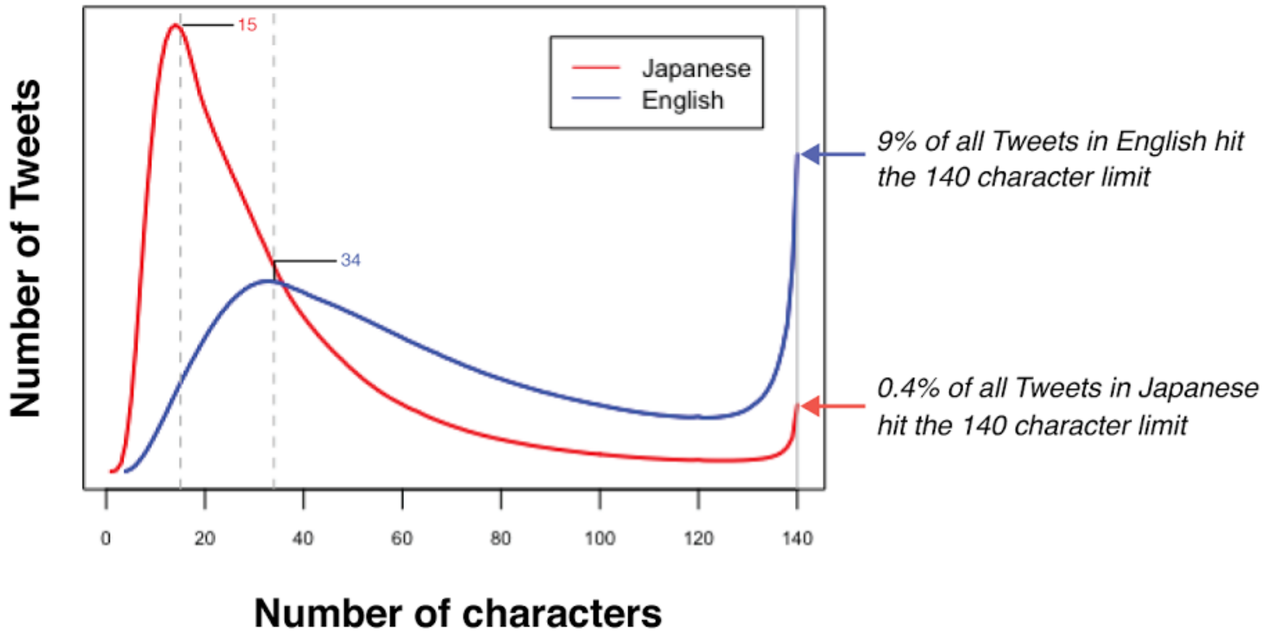


Nice "natural experiment" built into Twitter

https://blog.twitter.com/en_us/topics/product/2017/Giving-you-more-characters-to-express-yourself.html

A data science question

New question: if the limit is raised, will people just cram into that new length? What should the limit be?



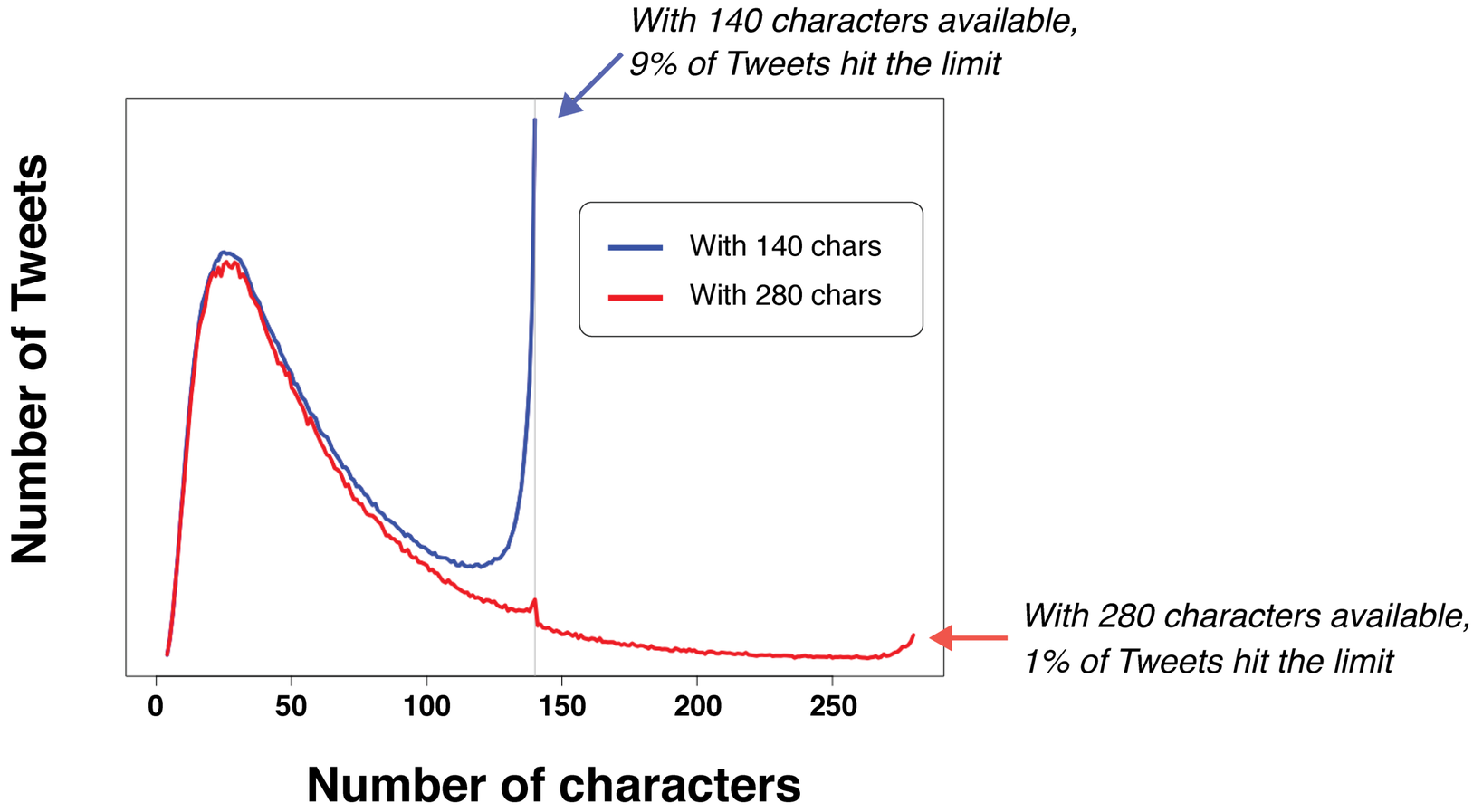
Nice "natural experiment" built into Twitter

https://blog.twitter.com/en_us/topics/product/2017/Giving-you-more-characters-to-express-yourself.html

A data science question

Twitter decided to double the limit to 280

The change was rolled out in 2017



https://blog.twitter.com/en_us/topics/product/2017/tweetingmadeeasier.html

A data science question

Notice the process

1. Ask question: *140 limit correct*?
   i. Why ask this question? Is it important? History/context?
   ii. Can we answer this? With data?
2. Hypothesize about what the data will tell us
3. Test the hypotheses
4. Insight → *English-vs-Japanese*
5. Decision: *double limit to 280*

Low-tech or High-tech?

Low-tech or High-tech?

Neural language representations predict outcomes of scientific research

James P. Bagrow[1,2,*], Daniel Berenberg[3,2], and Joshua Bongard[3,2]

[1]Department of Mathematics & Statistics, University of Vermont, Burlington, VT, United States
[2]Vermont Complex Systems Center, University of Vermont, Burlington, VT, United States
[3]Department of Computer Science, University of Vermont, Burlington, VT, United States
[*]Corresponding author. Email: james.bagrow@uvm.edu, Homepage: bagrow.com

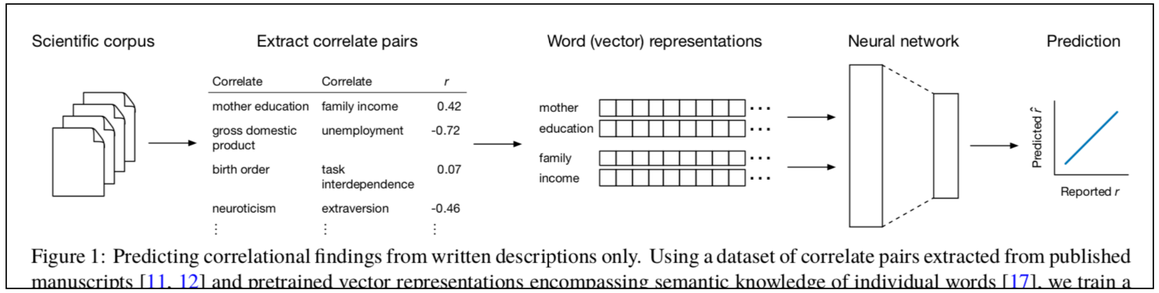


Figure 1: Predicting correlational findings from written descriptions only. Using a dataset of correlate pairs extracted from published manuscripts [11, 12] and pretrained vector representations encompassing semantic knowledge of individual words [17], we train a

Low-tech or High-tech?

Google Forms

Neural language representations predict outcomes of scientific research

James P. Bagrow[1,2,*], Daniel Berenberg[3,2], and Joshua Bongard[3,2]

[1]Department of Mathematics & Statistics, University of Vermont, Burlington, VT, United States
[2]Vermont Complex Systems Center, University of Vermont, Burlington, VT, United States
[3]Department of Computer Science, University of Vermont, Burlington, VT, United States
[*]Corresponding author. Email: james.bagrow@uvm.edu, Homepage: bagrow.com

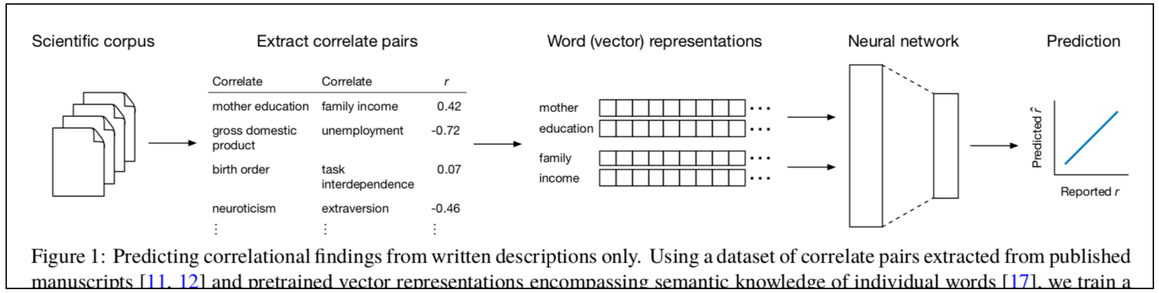


Figure 1: Predicting correlational findings from written descriptions only. Using a dataset of correlate pairs extracted from published manuscripts [11, 12] and pretrained vector representations encompassing semantic knowledge of individual words [17], we train a

Low-tech or High-tech?

Neural language representations predict outcomes of scientific research

James P. Bagrow[1,2,*], Daniel Berenberg[3,2], and Joshua Bongard[3,2]

[1]Department of Mathematics & Statistics, University of Vermont, Burlington, VT, United States
[2]Vermont Complex Systems Center, University of Vermont, Burlington, VT, United States
[3]Department of Computer Science, University of Vermont, Burlington, VT, United States
[*]Corresponding author. Email: james.bagrow@uvm.edu, Homepage: bagrow.com

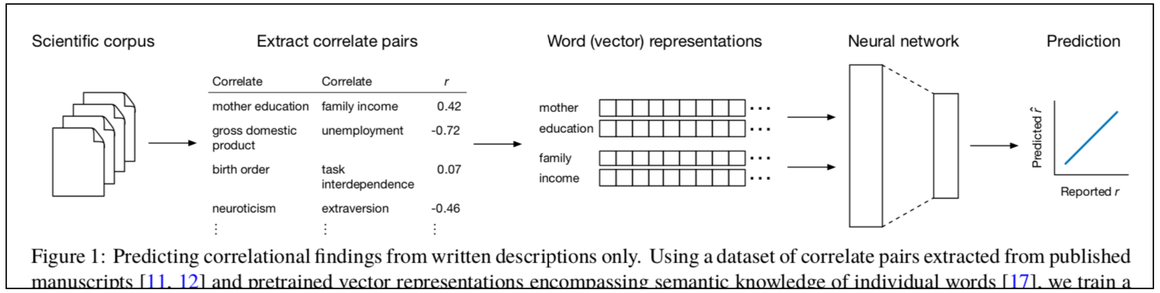
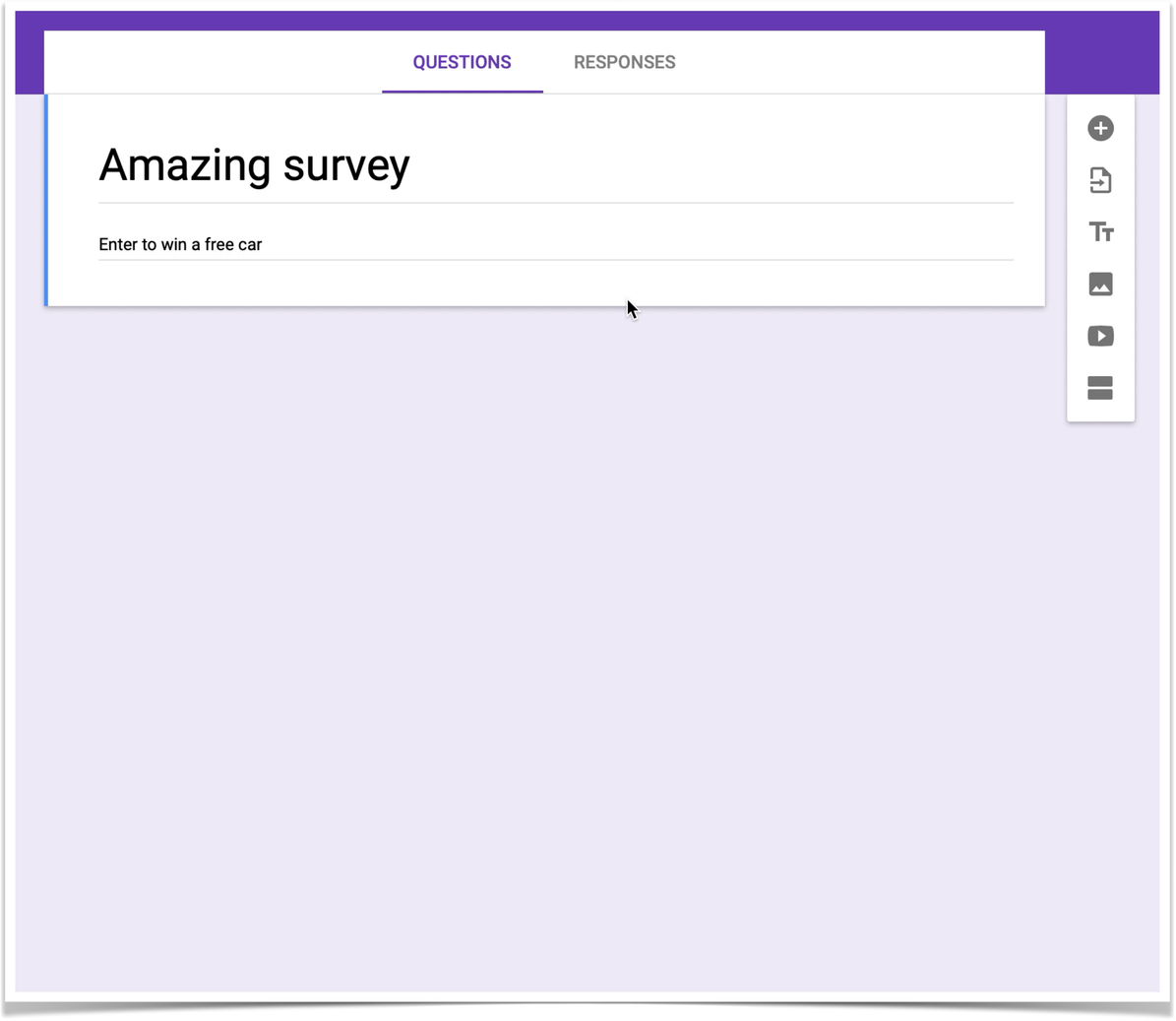
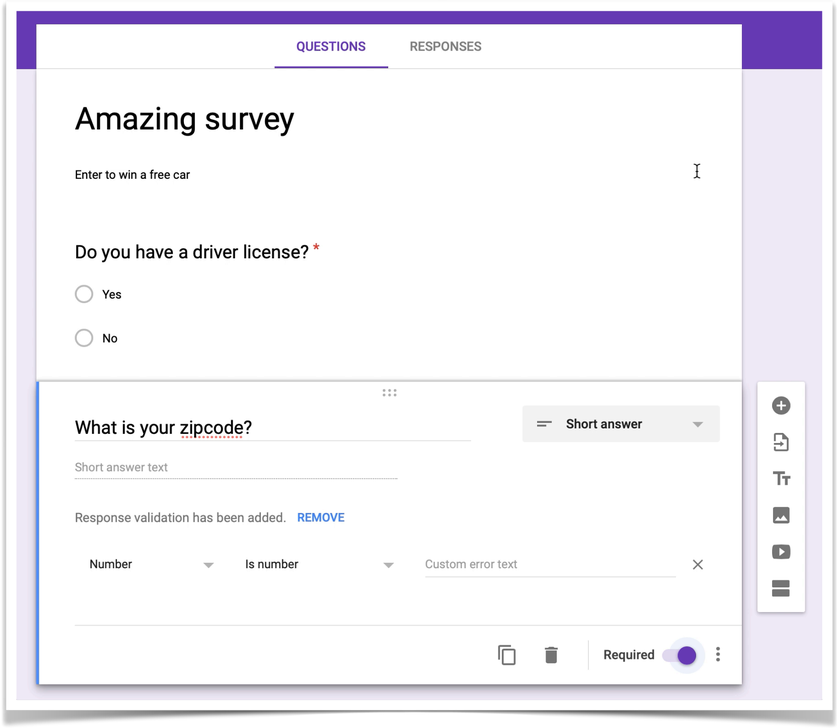


Figure 1: Predicting correlational findings from written descriptions only. Using a dataset of correlate pairs extracted from published manuscripts [11, 12] and pretrained vector representations encompassing semantic knowledge of individual words [17], we train a

Google Forms



Low-tech or High-tech?



Cloud AutoML BETA

Train high-quality custom machine learning models with minimal effort and machine learning expertise.

Try AutoML | View documentation

cloud.google.com/automl/

AutoML.org
Freiburg-Hannover

Democratizing AI: Non-expert design of prediction tasks

James P. Bagrow[1,2]

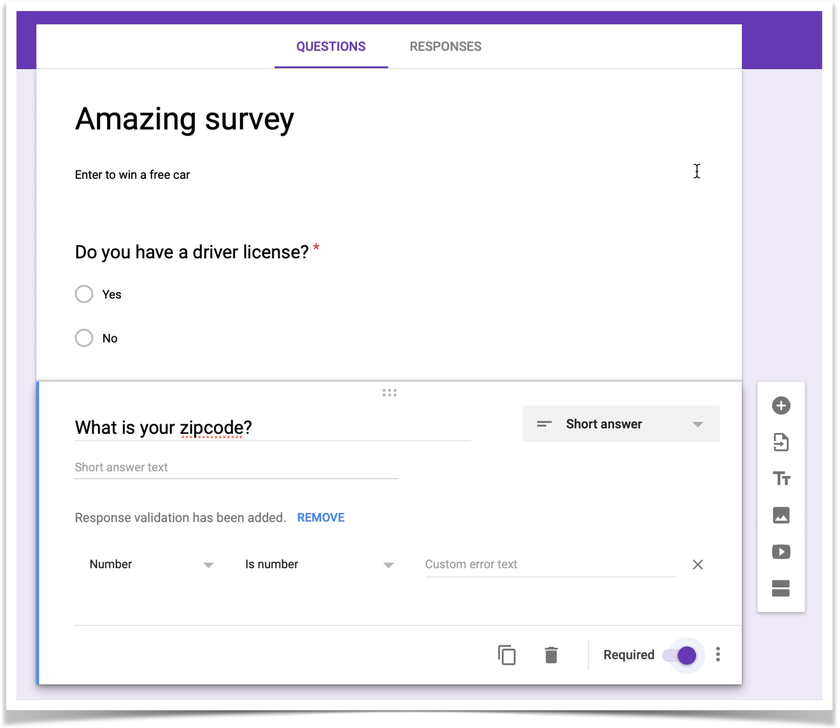
[1]**Department of Mathematics & Statistics, University of Vermont**
[2]**Vermont Complex Systems Center**

Corresponding author:
James P. Bagrow

Email address: james.bagrow@uvm.edu

bagrow.com/#pub50

Low-tech or High-tech?



Cloud AutoML BETA

Train high-quality custom machine learning models with minimal effort and machine learning expertise.

Try AutoML | View documentation

cloud.google.com/automl/

AutoML.org
Freiburg-Hannover

Democratizing AI: Non-expert design of prediction tasks

James P. Bagrow[1,2]

[1]**Department of Mathematics & Statistics, University of Vermont**
[2]**Vermont Complex Systems Center**

Corresponding author:
James P. Bagrow

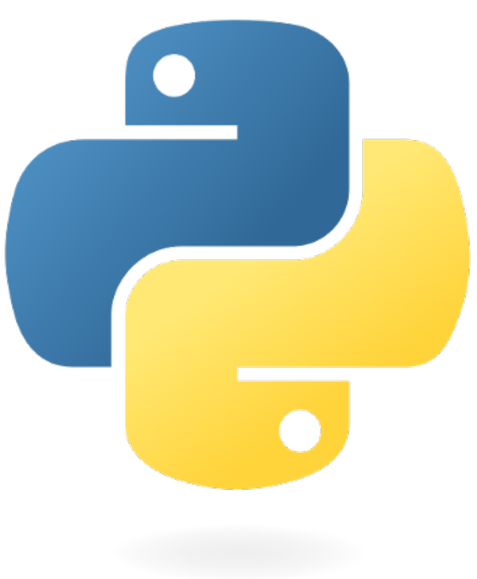
Email address: james.bagrow@uvm.edu

bagrow.com/#pub50

Today's Schedule

- ✓ Course logistics
- ✓ Intro + Motivation
- Intro/Review of programming

Intro/Review of programming



python™

This is a programming-intensive class. The language will be **Python**.

Why Python?

Why not:

R	Perl	STATA
MATLAB	Julia	⋮
IDL	SAS	?

Answers:

1. Popularity
2. Personal Preference
3. Plasticity

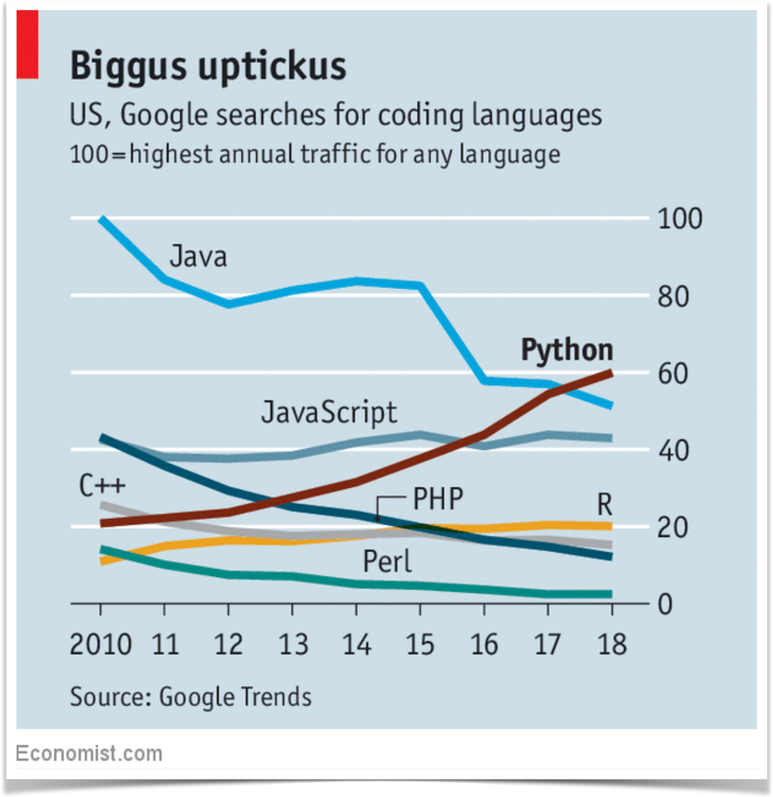
Why Python?

Why not:

R	Perl	STATA
MATLAB	Julia	⋮
IDL	SAS	?

Answers:

1. Popularity
2. Personal Preference
3. Plasticity

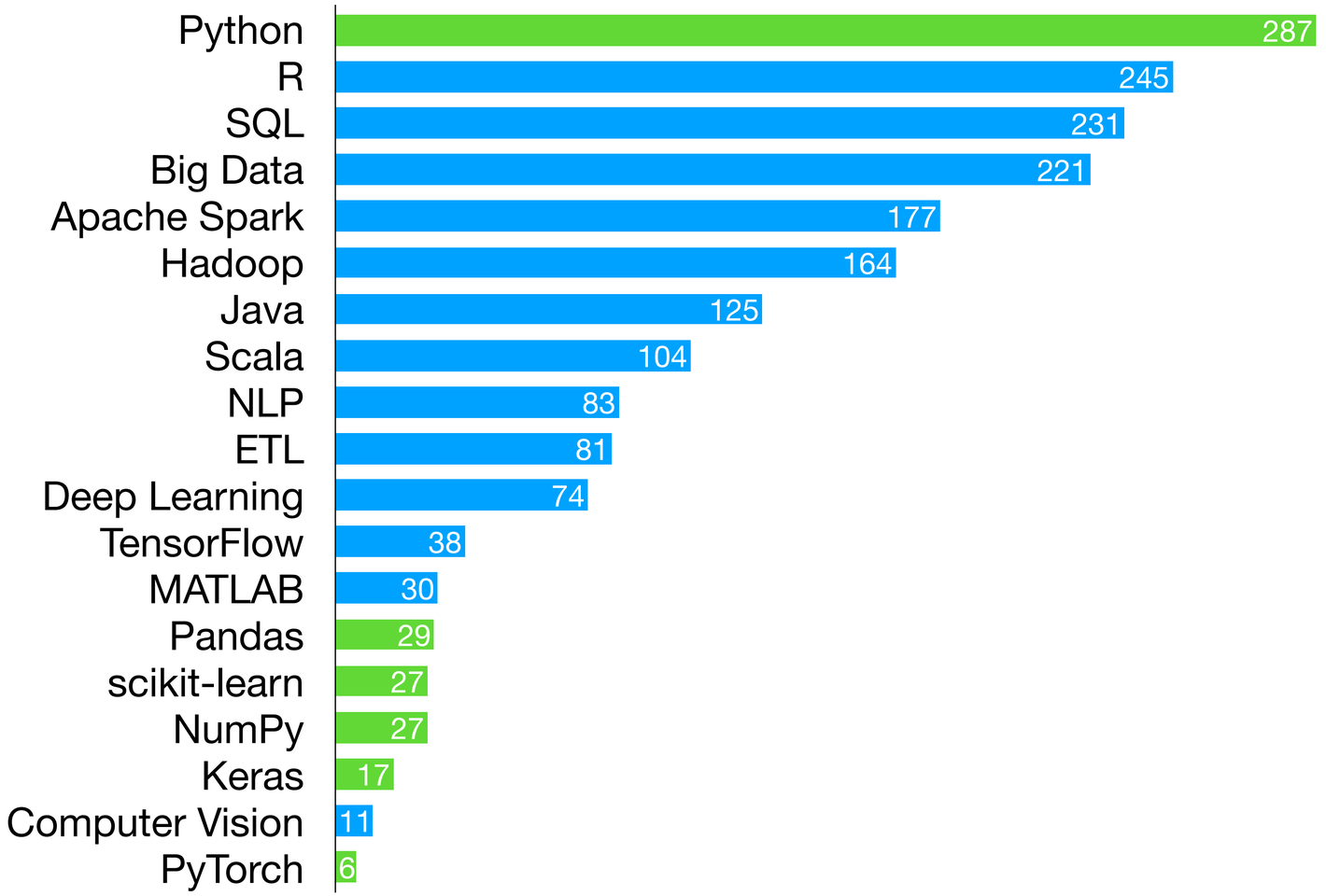


http://pypl.github.io/PYPL.html

Why Python?

Answers:
1. Popularity
2. Personal Preference
3. Plasticity

Skills mentioned in 300 **data scientist job postings** (June 2019)



https://cvcompiler.com/blog/how-to-become-more-marketable-as-a-data-scientist/

Programming review

```
print("Hello world")
```

This is a programming-intensive class. The language will be **Python**.

Programming review

Data types

`4` `3.45`

`“a”` `True`

Data structures

vectors, arrays, hashes, matrices, sets, etc.

Code

keywords `for`

`+` `-` `%` `#` `while`

Code structures

functions

libraries

modules

Programming review

Data types

`4` `3.45`

`"a"` `True`

Data structures

vectors, arrays, hashes, matrices, sets, etc.

Code

keywords `for`

`+` `-` `%` `#` `while`

Code structures

functions

libraries

modules

Programming review

Data types

`4` `3.45`

`"a"` `True`

Data structures

vectors, arrays, hashes, matrices, sets, etc.

Code

keywords `for`

`+` `-` `%` `#` `while`

Code structures

functions

libraries

modules

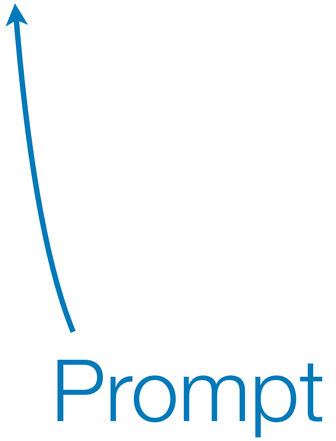
Objects/ Classes

Use Python interactively

```
Python 3.10.2 |Anaconda custom (x86_64)| (default, Aug  31 2021, 17:52:12)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Use Python interactively

```
Python 3.10.2 |Anaconda custom (x86_64)| (default, Aug  31 2021, 17:52:12)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



Use Python interactively

```
Python 3.10.2 |Anaconda custom (x86_64)| (default, Aug  31 2021, 17:52:12)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Use Python interactively

```
Python 3.10.2 |Anaconda custom (x86_64)| (default, Aug  31 2021, 17:52:12)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
```

Use Python interactively

```
Python 3.10.2 |Anaconda custom (x86_64)| (default, Aug  31 2021, 17:52:12)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
```

Use Python interactively

```
Python 3.10.2 |Anaconda custom (x86_64)| (default, Aug  31 2021, 17:52:12)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.2.79)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
```

As part of your homework and setup, you will learn about a much more advanced Python prompt called IPython

Use Python files

awesome_science.py:

```
import sys # not used

def product(x,y):
    return x*y

print("x =", product(5,7))
```

```
x = 35
```

Simple syntax

```
x = 34              # a comment
y = "Hello"
z = 3.45

if z == 3.45 or y == "Hello":
   x = x+1
   y = y + " World" # String concat.

print(x)
print(y)
```

Data types

integers	`x = 5`
floats	`pi = 3.14`
strings	`s = "hi\namy"`
booleans	`found = True`

Data structures

Lists

```
nums = [4,7,12,9]
names = ["John","Alice","Bob"]
[5, [1,2,3], "nested and mixed"]
```

Data structures

Example:
Using lists

```
>>> nums = [4,7,12,9]
>>> print(nums[0], nums[-1])
4 9
>>> print(nums[:3])
[4,7,12]
>>> print(nums[2:])
[12,9]
>>> nums.append(15)
>>> print(nums)
[4,7,12,9,15]
```

index access

slicing

modifying

Data structures

OK, but so what?

- Suppose I want to study a social network
- For each person, I have a list of their friends:

Data structures

OK, but so what?

- Suppose I want to study a social network
- For each person, I have a list of their friends:

$$\mathbf{LF} = [\mathbf{F}_1, \mathbf{F}_2, \ldots]$$

Data structures

OK, but so what?

- Suppose I want to study a social network
- For each person, I have a list of their friends:

$$\mathbf{LF} = [\mathbf{F}_1, \mathbf{F}_2, \ldots]$$

```
>>> LF = [["Alice", "John", "Bob"],
          ["John", "Bob", "Stacy"],…
         ]
>>> print(LF[1])
["John", "Bob", "Stacy"]
```

Data structures

OK, but so what?

- Suppose I want to study a social network
- For each person, I have a list of their friends:

$$\mathbf{LF} = [\mathbf{F}_1, \mathbf{F}_2, \ldots]$$

```
>>> LF = [["Alice", "John", "Bob"],
          ["John", "Bob", "Stacy"],…
         ]
>>> print(LF[1])
["John", "Bob", "Stacy"]
```

But whose friends are these? Who is person "1"?

Data structures

Dictionaries

A list maps the integers 0, 1, … to some values.
Why limit ourselves to integers?

Data structures

Dictionaries

A list maps the integers 0, 1, … to some values.
Why limit ourselves to integers?

```
>>> D = {"Gwen":["Alice", "John", "Bob"],
         "Peter":["John", "Bob", "Stacy"]}
```

Data structures

Dictionaries

A list maps the integers 0, 1, … to some values.
Why limit ourselves to integers?

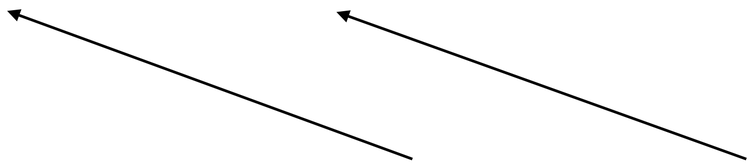
```
>>> D = {"Gwen":["Alice", "John", "Bob"],
         "Peter":["John", "Bob", "Stacy"]}
>>> print(D['Gwen'])
['Alice', 'John', 'Bob']
```

Data structures

Dictionaries

A list maps the integers 0, 1, … to some values.
Why limit ourselves to integers?

```
>>> D = {"Gwen":["Alice", "John", "Bob"],
         "Peter":["John", "Bob", "Stacy"]}
```



Generalization: maps **KEYS** to **VALUES**

Data structures

Dictionaries

A list maps the integers 0, 1, … to some values.
Why limit ourselves to integers?

```
>>> user2friends = {"Gwen":["Alice", "John", "Bob"],
                    "Peter":["John", "Bob", "Stacy"]}
>>> print(user2friends['Gwen'])
['Alice', 'John', 'Bob']
```

Data structures

Dictionaries

A list maps the integers 0, 1, … to some values.
Why limit ourselves to integers?

```
>>> user2friends = {"Gwen":["Alice", "John", "Bob"],
                    "Peter":["John", "Bob", "Stacy"]}
>>> print(user2friends['Gwen'])
['Alice', 'John', 'Bob']
```


```
>>> print(D['Gwen'])
```

Data structures

Dictionaries

A list maps the integers 0, 1, … to some values.
Why limit ourselves to integers?

```
>>> {"Gwen" :["Alice", "John", "Bob"],
     "Peter":["John", "Bob", "Stacy"]}
```

Keys must be **unique** and **immutable**

Finding a key in a dict is **very fast** (make great lookup tables)

Data structures

Dictionaries cont.

```
>>> D = {} # empty
>>> D[5] = 9
>>> D[3] = 12
>>> print(D)
{3: 12, 5: 9}
>>> D[5] = 8
>>> del D[3]
>>> print(D)
{5: 8}
```

dicts are **unordered**

replaced a value

deleted a (key,value)

Data structures

Dictionaries cont.

```
>>> D = {} # empty
>>> D[5] = 9
>>> D[3] = 12
>>> print(D)
{3: 12, 5: 9}
```

dicts are **unordered**

More data structures to come

```
>>> print(D)
{5: 8}
```

Control flow | for loops

A Python for-loop is like a "for each" loop:

The looping variable (**x**) **becomes *each*** item in the list

```
L = ["Alice","Bob"]

for x in L:
  print("x =", x)
---
```

Also: white space is **significant**!

Control flow | for loops

A Python for-loop is like a "for each" loop:

The looping variable (**x**) **becomes *each*** item in the list

```
L = ["Alice","Bob"]

for x in L:
  print("x =", x)
```

Output:

```
x = Alice
x = Bob
```

Control flow | for loops

A Python for-loop is like a "for each" loop:

The looping variable (**x**) **becomes *each*** item in the list

```
L = ["Alice","Bob"]

for x in L:
  print("x =", x)
```

Compare to a *C-style* for loop:

```
for (i=0; i<2; i++) {
   printf("%s\n", L[i]);
}
```

Control flow | for loops

```
L = ["Alice","Bob"]
```

compare:

```
for x in L:
    print("x =", x)
```

versus:

```
for i in range(len(L)):
    print("x =", L[i])
```

Control flow | for loops

```
L = ["Alice","Bob"]
```

✓ compare:

```
for x in L:
    print("x =", x)
```

✘ versus:

```
for i in range(len(L)):
    print("x =", L[i])
```

Becoming *comfortable* with "for each" loops:
→ less and more readable code!
→ fewer mistakes!

Control flow | for loops

One temptation is multiple "iterables":

```
X = […]
Y = […]
for i in range(len(X)):
   print("s =", X[i]+Y[i])
```

Control flow | for loops

One temptation is multiple "iterables":

```
X = […]
Y = […]
for i in range(len(X)):
   print("s =", X[i]+Y[i])
```

✘

Instead, embrace new tools. For example, use **`zip`**:

```
for x,y in zip(X,Y):
   print("s =", x+y)
```

✓

More "Pythonic" code

Control flow

If statements

```
x = 5
y = "yes"
z = True
p = 0.333

if x < 10 and y == "yes" and not z:
   print("won't happen")
print("if statement over")


if 0 < p < 1:
    print("yep")
```

nice! same as:

```
p > 0 and p < 1
```

Control flow

While loops

```
x = 0
while True:
   x += 1
   if x >= 100:
      break
print(x)
```

```
x = 0
while x < 100:
   x += 1
print(x)
```

(`x += 1` is a shortcut for `x = x + 1`)

Functions

```
def my_function(x, y):
    """This is the docstring.

    This is my function. There are many
    like it but this one is mine.
    """
    # The code would go here..
    pass


def second_func(username, password=None):
    if password is not None:
        print("found a password")
```

Colors!

(syntax highlighting)

Functions

```
def my_function(x, y):
    """This is the docstring.

    This is my function. There are many
    like it but this one is mine.
    """
    # The code would go here..
    pass
```

Colors!
(syntax highlighting)

```
def second_func(username, password=None):
    if password is not None:
        print("found a password")
```

default value for
optional argument

"Batteries included"

Want to write a web crawler? No problem!

To download web pages:

"Batteries included"

Want to write a web crawler? No problem!

To download web pages:

```
>>> import urllib
>>> web = urllib.request.urlopen("http://bagrow.com/ds1")
>>> text = web.read() # read?
>>> print(text[2800:3400])
```

"Batteries included"

Want to write a web crawler? No problem!

To download web pages:

```
>>> import urllib
>>> web = urllib.request.urlopen("http://bagrow.com/ds1")
>>> text = web.read() # read?
>>> print(text[2800:3400])
```

```
b'data,\nso it is more important than ever to understand how to collect,
process, and\nanalyze these data. A picture is worth a thousand words, so
visualizations,\nfrom scientific plots and infographics to interactive data
explorers, are\ncrucial to summarize and communicate new discoveries.\n</
p>\n\n</div><!--/row-->\n\n\n\n<div class="row">\n    <h3>Resources</h3>\n
<div class="col-md-7">\n    <ul>\n        <li>The <b><a
href="syllabus_stat287_fall2016.pdf">course syllabus</a></b>. Be sure to check
this out. </li>\n        <li>The course introductory reading, <a
href="whirlwindtourpython/00-Title.html">A W'
```

"Batteries included"

Want to write a web crawler? No problem!

Programmatic power:

```
import urllib

website = "http://uvm.edu/"
names = ["alice","bob","john"]

for name in names:
    url = website + name + ".html"
    print(url)
```

```
http://uvm.edu/alice.html
http://uvm.edu/bob.html
http://uvm.edu/john.html
```

"Batteries included"

Want to write a web crawler? No problem!

Programmatic power:

```
import urllib

names = ["alice","bob","john"]

for name in names:
    url = "http://uvm.edu/{}.html".format(name)
```

Handy reference:
https://pyformat.info

"Batteries included"

Want to write a web crawler? No problem!

Programmatic power:

```
import urllib

names = ["alice","bob","john"]

for name in names:
    url = "http://uvm.edu/{}.html".format(name)
```

✔ *

string substitution

Better than using + to build strings

Handy reference: https://pyformat.info

* Even cooler: f-strings! → `url = f"http://uvm.edu/{name}.html"`

Learning resources

A Whirlwind Tour of Python for STAT/CS 287

James Bagrow

This mini-text serves as introductory reading for my course, Data Science I, held at the University of Vermont. It has been very lightly adapted and specialized from the original "A Whirlwind Tour of Python" by Jake VanderPlas, and he deserves most of the credit for this work.

A Whirlwind Tour of Python is a fast-paced introduction to essential components of the Python language for researchers and developers who are already familiar with programming in another language.

The material is particularly aimed at those who wish to use Python for data science and/or scientific programming, and in this capacity serves as an introduction to Jake VanderPlas' book, The Python Data Science Handbook.

Contents

DS1 Lecture 01 supplement

Jim Bagrow

This notebook acts as an additional supplement on programming in Python. You are expected to learn this material durin quiz of the semester will assess this.

This document complements the course's **"Whirlwind Tour" reading**. You are also expected to complete this reading as this supplement after reading the whirlwind tour.

About this document

- This is a **Jupyter Notebook**. I'll be using these for code-heavy lectures and lecture supplements. Notebooks contair in a document, interleaved with narrative (non-code) text.
- Your **projects and assignments** will be submitted as Python scripts (`.py` files).

Contents:

Computer setup

We're using a scientific Python environment called **Anaconda**

Supports Windows/Mac/Linux

Free for academic use

Includes **package manager**, text editor, and interactive prompt

https://www.anaconda.com/download/

Today's Schedule

✓ Intro + Motivation

✓ Course logistics

✓ Intro/Review of programming

Next time: Intro/Review of prob/stat

Homework ⟶

Homework (1 of 2)

1. Register and install Anaconda Python on your machine (see Python Setup slides)
→ Complete **HW01**
→ HW01 includes additional setup reading

2. Work through **reading**—there's a lot but the more prepared you are now, the faster you can work later

(Quiz 01 will assess your preparation following this reading)

con't ⟶

Homework (2 of 2)

Python setup

Jim Bagrow
STAT/CS 287 — Data Science 1

Preparing and submitting assignments

Jim Bagrow
STAT/CS 287 —Data Science 1

Homework (2 of 2)

Python setup

Jim Bagrow
STAT/CS 287 — Data Science 1

Preparing and submitting assignments

Jim Bagrow
STAT/CS 287 —Data Science 1

A Whirlwind Tour of Python for STAT/CS 287

James Bagrow

This mini-text serves as introductory reading for my course, Data Science I, held at the University of Vermont. It has been very lightly adapted and specialized from the original "A Whirlwind Tour of Python" by Jake VanderPlas, and he deserves most of the credit for this work.

A Whirlwind Tour of Python is a fast-paced introduction to essential components of the Python language for researchers and developers who are already familiar with programming in another language.

The material is particularly aimed at those who wish to use Python for data science and/or scientific programming, and in this capacity serves as an introduction to Jake VanderPlas' book, The Python Data Science Handbook.

Contents

DS1 Lecture 01 supplement

Jim Bagrow

This notebook acts as an additional supplement on programming in Python. You are expected to learn this material durin quiz of the semester will assess this.

This document complements the course's **"Whirlwind Tour" reading**. You are also expected to complete this reading as this supplement after reading the whirlwind tour.

About this document

- This is a **Jupyter Notebook**. I'll be using these for code-heavy lectures and lecture supplements. Notebooks contair in a document, interleaved with narrative (non-code) text.
- Your **projects and assignments** will be submitted as Python scripts (`.py` files).

Contents: