Today’s Schedule

Intro + Motivation

Course logistics

Intro/Review of programming
Homework

Register and install Anaconda on your machine
  ➞ Complete HW00.
  ➞ HW00 includes additional setup reading

Work through reading

More Python review next time
Followed by review of probability and statistics
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?

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

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


How much money you earn in the 'sexiest job of the 21st century ... www.businessinsider.com/how-much-money-you-earn-in-the-sexiest-jo... Business Insider
What is data science?

mixed with **domain knowledge**:

- social networks/sociology
- biology or drug discovery
- neuroscience
- business
- etc…
What is data science?

A mixed course is incredibly useful for majors preparing for today’s interdisciplinary research and industry.

“All the skills I wish I learned before grad school!”

- Me, now
What is data science?

What research and coursework make data science look like

- Data collection, preprocessing, filtering
- Statistical analysis, predictive modeling, machine learning
- Presentation and visualization, reporting
What is data science?

What research and coursework make data science look like:

- Data collection, preprocessing, filtering
- Statistical analysis, predictive modeling, machine learning
- Presentation and visualization, reporting

What data science actually is:

- Data collection, preprocessing, filtering
- Statistical analysis, predictive modeling, machine learning
- Presentation and visualization, reporting, explanation
What is data science?

A **meta-field** field encompassing:

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

:
Course logistics

Website: http://bagrow.com/ds1/

Syllabus
Lecture notes/slides

Homeworks, Projects on Blackboard

Let’s go over the syllabus right now!
Topics include

• "good" practices for scientific computing
• practical data exploration and analysis
• web access to data sets
• Natural language processing (NLP) and ML
• Classic regression and inference methods
• A/B testing and multiple comparisons problems
• bayesian statistics and computation
• visualization principles and design
• Interpreting and communicating results!
This is a programming-intensive class. The language will be **python**.
Why Python?  

Why not: R  
MATLAB  
IDL  
Perl  
Julia  
SAS  
STATA
Why Python?  Why not: R  
MATLAB  
IDL  
Perl  
Julia  
SAS  
STATA  

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

http://pypl.github.io/PYPL.html
Why Python? Why not: R, MATLAB, IDL, Perl, Julia, SAS, ...?

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

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

print("Hello world")

This is a programming-intensive class. The language will be python.
Programming review

Data types
Programming review

Data types

4  3.45

“a”  True
# Programming review

<table>
<thead>
<tr>
<th>Data types</th>
<th>Data structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 3.45</td>
<td></td>
</tr>
<tr>
<td>“a”  True</td>
<td></td>
</tr>
</tbody>
</table>
Programming review

Data types

4 3.45

“a” True

Data structures

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

<table>
<thead>
<tr>
<th>Data types</th>
<th>Data structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 3.45</td>
<td>vectors, arrays, hashes, matrices, sets, etc.</td>
</tr>
<tr>
<td>“a” True</td>
<td></td>
</tr>
</tbody>
</table>

Code
Programming review

Data types

4 3.45
“a” True

Data structures

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

Code

keywords for

+ - % # while
Programming review

Data types

4  3.45
“a”  True

Code

keywords  for
+  -  %  #  while

Data structures

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

Code structures
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

Code
- keywords
- for
- + - % #
- while

Data structures
- vectors, arrays, hashes, matrices, sets, etc.

Code structures
- functions
- libraries
- modules

Objects/Classes
Python has all of these
Use it interactively

Python 3.5.2 |Anaconda custom (x86_64)| (default, Jul 2 2016, 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 it interactively

Python 3.5.2 |Anaconda custom (x86_64)| (default, Jul 2 2016, 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.

>>> Prompt
Use it interactively

Python 3.5.2 |Anaconda custom (x86_64)| (default, Jul 2 2016, 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 it interactively

Python 3.5.2 |Anaconda custom (x86_64)| (default, Jul 2 2016, 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 it interactively

Python 3.5.2 |Anaconda custom (x86_64)| (default, Jul 2 2016, 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 it with files

script.py:
Use it with files

script.py:

```python
import sys # not used

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

print("x = ", product(5, 7))
```
Use it with files

script.py:

```python
import sys  # not used

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

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

x = 35
Simple syntax

```python
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\tmom"

booleans
found = True
Data structures

Lists
Data structures

Lists

nums = [4, 7, 12, 9]
Data structures

Lists

nums = [4,7,12,9]
names = ["John","Alice","Bob"]
Data structures

Lists

ums = [4, 7, 12, 9]

names = ['John', 'Alice', 'Bob']

[5, [1, 2, 3], 'nested and mixed']
Using lists

```python
>>> 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
OK, so what?

I want to study a social network

```python
>>> F = ['Alice', 'John', 'Bob']
```
OK, so what?

I want to study a social network

```python
>>> F = ['Alice', 'John', 'Bob']
```

But whose friends are these?
OK, so what?

I want to study a social network

```python
>>> F = ["Alice", "John", "Bob"]

But whose friends are these?

```python
>>> L = [["Alice", "John", "Bob"],
      ["John", "Bob", "Stacy"]]
```python

```python
>>> print(L[1])
["John", "Bob", "Stacy"]
```
OK, so what?

I want to study a social network

>>> F = ["Alice", "John", "Bob"]

But whose friends are these?

>>> L = [["Alice", "John", "Bob"],
       ["John", "Bob", "Stacy"]]

>>> print(L[1])
["John", "Bob", "Stacy"]

But who is person "1"?
Dictionaries

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

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

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

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

```python
>>> F = {
    "Gwen": ["Alice", "John", "Bob"],
    "Peter": ["John", "Bob", "Stacy"]
}
>>> print(F["Gwen"])
```
Dictionaries

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

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

A list maps numbers 0, 1, ... to values. Why limit ourselves to integers?

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

Generalization: maps **KEYS** to **VALUES**
Dictionaries cont.

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

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

- Keys must be **unique** and **immutable**
Dictionaries cont.

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

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

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

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

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

```python
>>> 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
Dictionaries cont

```python
>>> 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
Dictionaries cont

```python
>>> 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)
More data structures to come
Control flow

L = ["Alice","Bob"]

for x in L:
    print("x =", x)
Control flow

L = ["Alice","Bob"]

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

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)

A python for-loop is like a "for each" loop:
The looping variable (x) becomes each item in the list

White space is significant!
Control flow

L = ['Alice', 'Bob']

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

Output:

x = Alice
x = Bob
Control flow

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

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

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

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 makes for less code and more readable code!
Control flow

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

If statements
Control flow

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

If statements

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

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

If statements

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

if 0 < p < 1:
    print("yep")
Control flow

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:  # nice! same as:
    print("yep")
    p > 0 and p < 1
Control flow

While loop
Control flow

While loop

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

(x += 1 is a shortcut for x = x + 1)
Control flow

While loop

\[
x = 0
while \text{True}:
    x += 1
    if x >= 100:
        break
print(x)
\]

\[
x = 0
while x < 100:
    x += 1
    print(x)
\]

\(\text{(x += 1 is a shortcut for x = x + 1)}\)
Functions

def my_function(x, y):
    """This is the docstring. This function does blah blah blah.
    """
    # The code would go here..
    pass

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

```python
>>> 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:

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

b'data,
so it is more important than ever to understand how to collect, process, and analyze these data. A picture is worth a thousand words, so visualizations, from scientific plots and infographics to interactive data explorers, are crucial to summarize and communicate new discoveries.

</p>

</div><!--/row-->

</div><!--/row-->

<div class="row">
    <h3>Resources</h3>

    <div class="col-md-7">
        <ul>
            <li>The <b><a href="syllabus_stat287_fall2016.pdf">course syllabus</a></b>. Be sure to check this out. </li>
            <li>The course introductory reading, <a href="whirlwindtourpython/00-Title.html">A W'
```
"Batteries included"

Programmatic power:
"Batteries included"

Programmatic power:

```python
import urllib

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

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

Programmatic power:

```python
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"

Programmatic power:

```python
import urllib

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

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

Programmatic power:

```python
import urllib

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

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

string substitution
A Whirlwind Tour of Python for STAT/CS 287

James Bagrow, August 2016

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

1 Introduction .......... 3
   1.1 About the authors ................................... 4
   1.2 Installation and Practical Considerations .......... 4

2 How to Run Python Code .......... 5

3 A Quick Tour of Python Language Syntax .......... 7
   3.1 Comments Are Marked by # ................................... 8
   3.2 End-of-Line TERMINATES a Statement .......... 8
   3.3 Semicolons Can OptionallY Terminate a Statement .......... 9
   3.4 Indentation: Whitespace Matters! .......... 9
   3.5 Whitespace Within Lines Does Not Matter .......... 10
Computer setup

We're using a scientific python environment called **Anaconda**
Computer setup

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

Free for academic use

Supports Windows/Mac/Linux

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

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

Let's see it in action
Homework

Register and install Anaconda on your machine
→ Complete HW00.
→ HW00 includes additional setup reading

Work through reading

More Python review next time
Followed by review of probability and statistics