

Introduction to Data Science with Python

MIT Political Science Methods Workshop

Soubhik Barari
Computational and Statistical Research Specialist
MIT Political Methodology Lab

February 9 2018

Pre-requisites:

- 1 Proficiency in R
- 2 Proficiency w/data analysis

Workshop materials (setup, slides, notebooks):

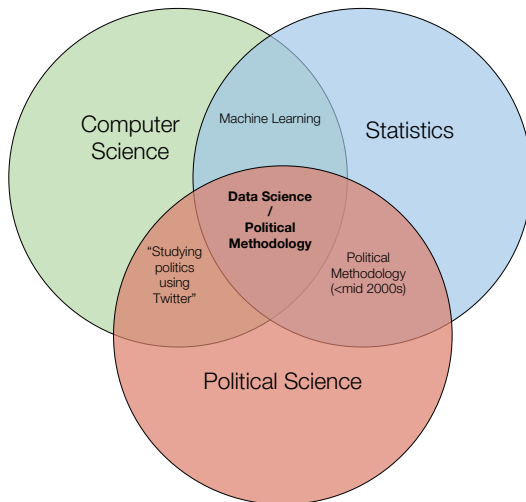
<https://github.com/soubhikbarari/MITPolMeth-PythonDataSci-02-2018>

Roadmap

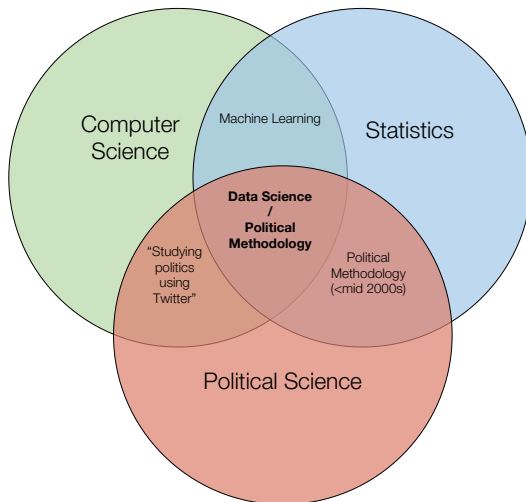
- 1 Overview [**slides**]
- 2 Python Fundamentals [**notebook**]
- 3 Python for Data Science [**notebook**]

Overview

Overview of Data Science



Overview of Data Science



***Not pictured in diagram:** your award-winning, tenure-track-faculty-position-garnering thesis dataset!

- **“Inferring Roll Call Scores from Campaign Contributions Using Supervised Machine Learning”** (Adam Bonica, 2017)
- **“Dynamic Estimation of Latent Opinion Using a Hierarchical Group-Level IRT Model”** (Devin Caughey et al., 2015)
- **“Measuring Trade Profiles with Two Billion Observations of Product Trade”** (Kosuke Imai et al., 2017)
- **“A New Automated Redistricting Simulator Using Markov Chain Monte Carlo”** (Ben Fifield et al., 2018)

Why use Python for Data Science

Because it...

- ① is easy to dig into lower levels (if you want).
- ② is able to do *many* different things.
- ③ is easy to learn (very accessible documentaion).
- ④ has *the best* general purpose machine learning toolkit.
- ⑤ is great for building re-usable things (vs. just analysis).

Overview of Python

Python is a programming language that is

① **Object-oriented**

```
model = library.CreateModel()  
model.fit(data)
```

② **Functional**

```
Y = map(lambda y: y**2, filter(lambda x: x < 5, X))
```

③ **Dynamically typed**

```
myVar = getNewData()
```

Overview of Python

Python is a programming language that is

① **Object-oriented** (organized!)

```
model = library.CreateModel()  
model.fit(data)
```

② **Functional** (clean!)

```
Y = map(lambda y: y**2, filter(lambda x: x < 5, X))
```

③ **Dynamically typed** (flexible!)

```
myVar = getNewData()
```

Python vs. R

Python relies more on packages.

R

```
df <- read.csv("data.csv")
```

Python

```
import pandas as pd  
df = pd.read_csv("data.csv")
```

Python vs. R

Python is more object-oriented.

R

```
fit <- lm(y ~ x, data=df)
predictions <- predict(fit, test)
```

Python

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(train["x"], train["y"])
predictions = lm.predict(test["x"])
```

Python vs. R

...however **Python** can do anything functional that **R** can in a simpler way (e.g. `lapply`, `vapply`, `shmap`... is just `map`!).

R

```
lapply(a_matrix, function(x) length(unique(x)))  
Filter(function(x) !is.numeric(x), a_matrix)
```

Python

```
map(lambda x: len(set(x)), a_matrix)  
filter(lambda x: type(x) != int, a_matrix)
```

Python vs. R

Python has better support for non-statistical tasks.

R

```
# Web-scraping basketball statistics

library(rvest)
page <- read_html(url)
table <- html_nodes(page, ".stats_table")[3]
rows <- html_nodes(table, "tr")
cells <- html_nodes(rows, "td a")
teams <- html_text(cells)

extractRow <- function(rows, i){
  if(i == 1){
    return
  }
  row <- rows[i]
  tag <- "td"
  if(i == 2){
    tag <- "th"
  }
  items <- html_nodes(row, tag)
  html_text(items)
}

scrapeData <- function(team) ...
```

Python

```
# Web-scraping basketball statistics

from bs4 import BeautifulSoup
import re

soup = BeautifulSoup(data, 'html.parser')
box_scores = []
for t in soup.find_all(id=re.compile ...
    rows = []
    for i, row in enumerate(t.find_all("tr")):
        if i == 0:
            continue
        elif i == 1:
            t = "th"
        else:
            t = "td"
        rd = [item.get_text() for item in ... ]
        rows.append(rd)
    box_scores.append(rows)
```

Python vs. R: Trade-offs

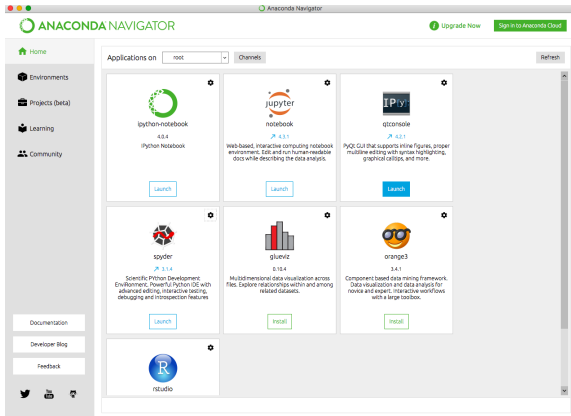
- Building tools vs. Doing analysis
- Flexibility vs. Convenience
- Speed vs. Parallelizability
- 'Computational' vs. 'Statistical'
- Great Machine Learning vs. Okay Machine Learning

Example Python Use Cases

- Build an end-to-end pipeline that automatically scrapes web data, runs analysis, and saves results.
- Write a slightly customized version of a standard machine learning algorithm using the `scikit-learn` framework.
- Work with (i.e. analyze, model, visualize) political text documents, audio data, images, or videos.

Running Python

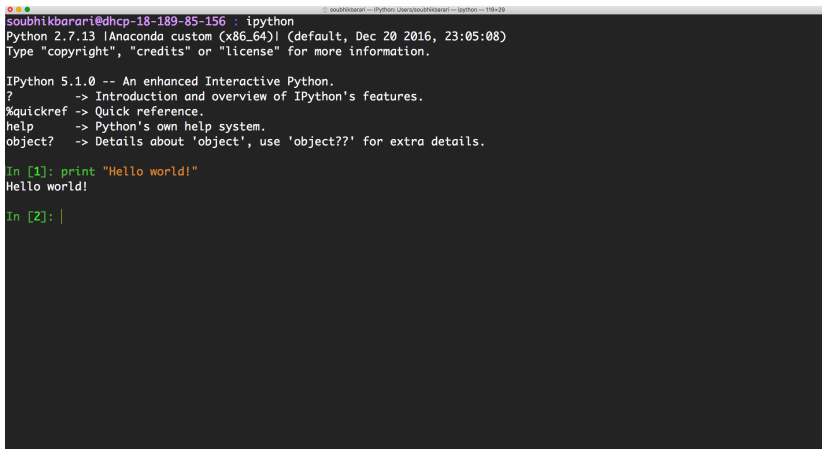
First, install the **Anaconda** distribution of Python**:



**Instructions can be found on setup.pdf at <https://github.com/soubhikbarari/MITPolMeth-PythonDataSci-02-2018>.

Running Python

A. Command line (Terminal / Command Prompt)



```
soubhikbarari@dhcp-18-189-85-156 : ipython
Python 2.7.13 |Anaconda custom (x86_64)| (default, Dec 20 2016, 23:05:08)
Type "copyright", "credits" or "license" for more information.

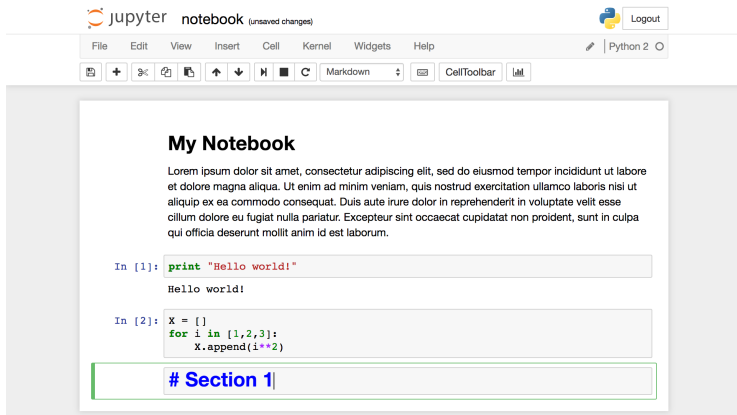
IPython 5.1.0 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?          -> Details about 'object', use 'object??' for extra details.

In [1]: print "Hello world!"
Hello world!

In [2]: |
```

Running Python

B. Notebook (Jupyter)



The screenshot displays the Jupyter Notebook web interface. At the top, the header includes the Jupyter logo, the text "jupyter notebook (unsaved changes)", a Python logo, and a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". A secondary bar shows "Python 2" and a refresh icon. A toolbar contains icons for saving, adding, deleting, and duplicating cells, as well as buttons for running, stepping through, and interrupting the kernel. The main content area is titled "My Notebook" and contains a paragraph of Lorem Ipsum text. Below the text are two code cells. The first cell, labeled "In [1]:", contains the code `print "Hello world!"` and has executed, showing "Hello world!". The second cell, labeled "In [2]:", contains a list comprehension: `X = []`, `for i in [1,2,3]:`, and `X.append(i**2)`. Below the code cells is a text cell containing the blue section header `# Section 1`. The bottom of the interface features a status bar with navigation icons and a search icon.

jupyter notebook (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Python 2

Save Add Delete Duplicate Run Step Through Interrupt Markdown Cell Toolbar

My Notebook

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

```
In [1]: print "Hello world!"
```

Hello world!

```
In [2]: X = []
        for i in [1,2,3]:
            X.append(i**2)
```

```
# Section 1
```

Running Python

C. IDE (Spyder)

The screenshot displays the Spyder IDE interface. The main editor window shows a Python script named `feature_atrap_tall.py` with the following code:

```
73 #
74 #
75 # ----- MLN -----
76 #
77 #
78 #
79 #####
80 # load data #
81 #####
82 #
83 # p = pd.read_pickle('E:\ARIMA\DATASET_PML')
84 # print "OK reading in data."
85 #
86 #####
87 # preprocess data #
88 #####
89 #
90 # for txt_elem in ["url", "link_text", "page_title", \
91 #                 "pdf_filename", "pdf_reader_text"]
92 #     print "... preprocessing %s" % txt_elem
93 #     df[txt_elem] = df[txt_elem].apply(proc_func, 1)
94 #
95 # for txt_elem in ["tos", "hds"]
96 #     print "... preprocessing %s" % txt_elem
97 #     df[txt_elem] = df[txt_elem].apply(lambda s: proc_func(" ", join(s)), 1)
98 #
99 # print "OK preprocessing done."
100 #
101 #####
102 # analyze "distribution" #
103 #####
104 #
105 #####
```

The bottom panel shows the Python console with the following text:

```
NOTE: The Python console is going to be REMOVED in Spyder 3.2. Please start to migrate your work to the IPython console instead.

Python 2.7.13 (Anaconda custom (x86_64)) (default, Dec 28 2016, 22:05:08)
[GCC 4.7.2] Compatible Apple LLVM 6.0 (clang-600.0.57) on darwin
Type "help", "copyright", "credits" or "license()" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org

>>> if __name__ == '__main__':
>>>
>>>
```

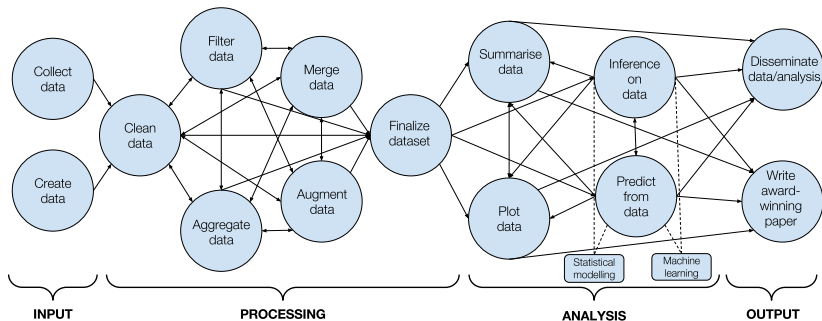
The right panel shows the Variable explorer and File explorer. The Variable explorer is empty. The File explorer shows the following files and folders:

Name	Size	Kind	Date Modified
anaconda		Folder	7/10/17 5:24 PM
data		Folder	8/11/17 10:52 AM
envs		Folder	8/14/17 12:17 PM
eval		Folder	7/10/17 5:23 PM
github_urls_ming		Folder	8/10/17 1:16 PM
jobs		Folder	7/11/17 6:38 PM
lib		Folder	7/28/17 4:00 PM
input		Folder	7/27/17 6:13 PM
perilized		Folder	8/14/17 12:00 PM
pipeline		Folder	12/8/17 4:43 PM

Python Fundamentals

(<https://github.com/soubhikbarari/MITPolMeth-PythonDataSci-02-2018/fundamentals.ipynb>)

The Data Science Pipeline



Python for Data Science

(<https://github.com/soubhikbarari/MITPolMeth-PythonDataSci-02-2018/datasci.ipynb>)

Other Jupyter notebooks:

- 1 Scientific Computing with Python:
<https://github.com/jrjohansson/scientific-python-lectures>
- 2 Python Data Science Handbook:
<https://github.com/jakevdp/PythonDataScienceHandbook>

At MIT:

- 1 MIT Libraries : Data Consultation Services
- 2 Harvard-MIT Data Center Cluster
- 3 XVII : MIT Political Methodology Lab Computing Cluster
- 4 sbarari@mit.edu