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

Prologue

Pre-requisites:

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

Workshop materials (setup, slides, notebooks):

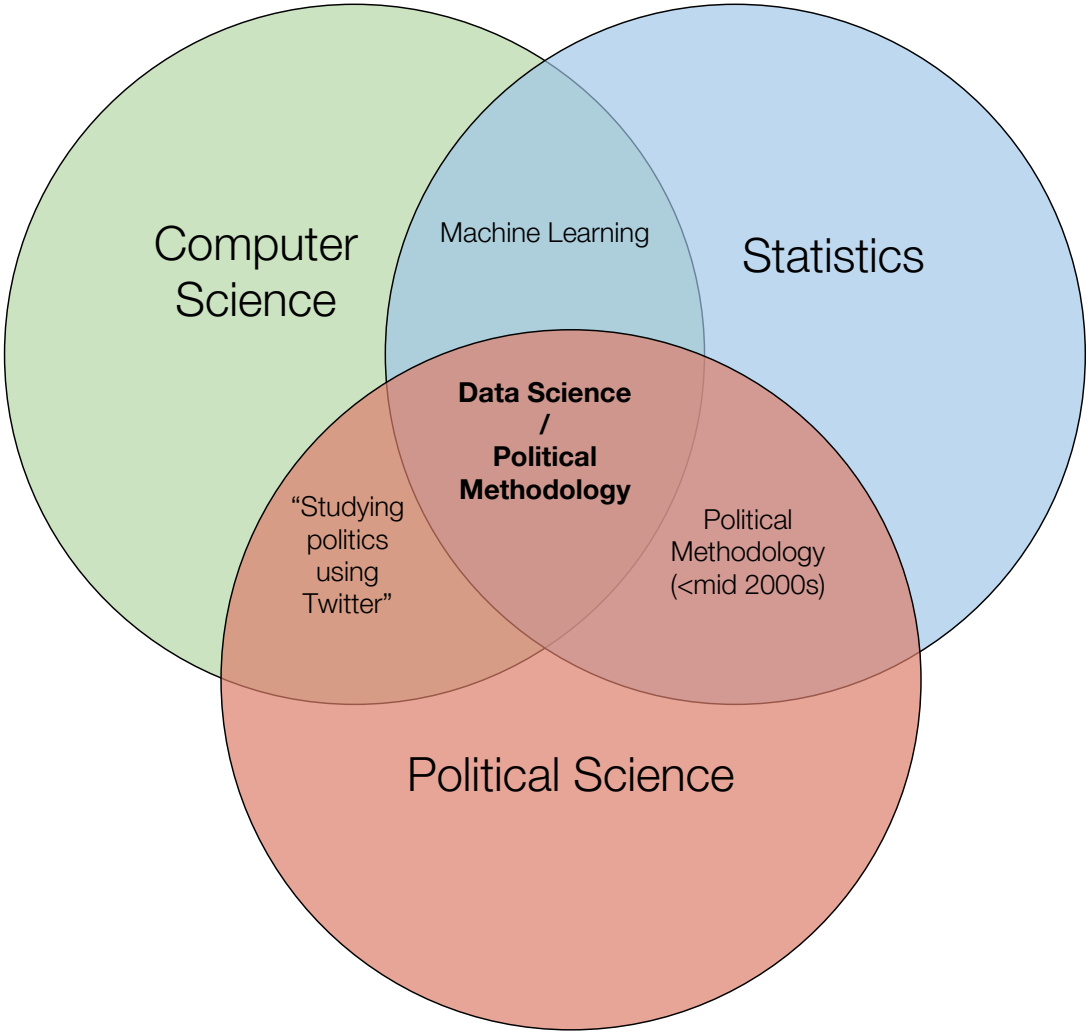
`https://github.com/soubhikbarari/MITPolMeth-PythonDataSci-02-2018`

Roadmap

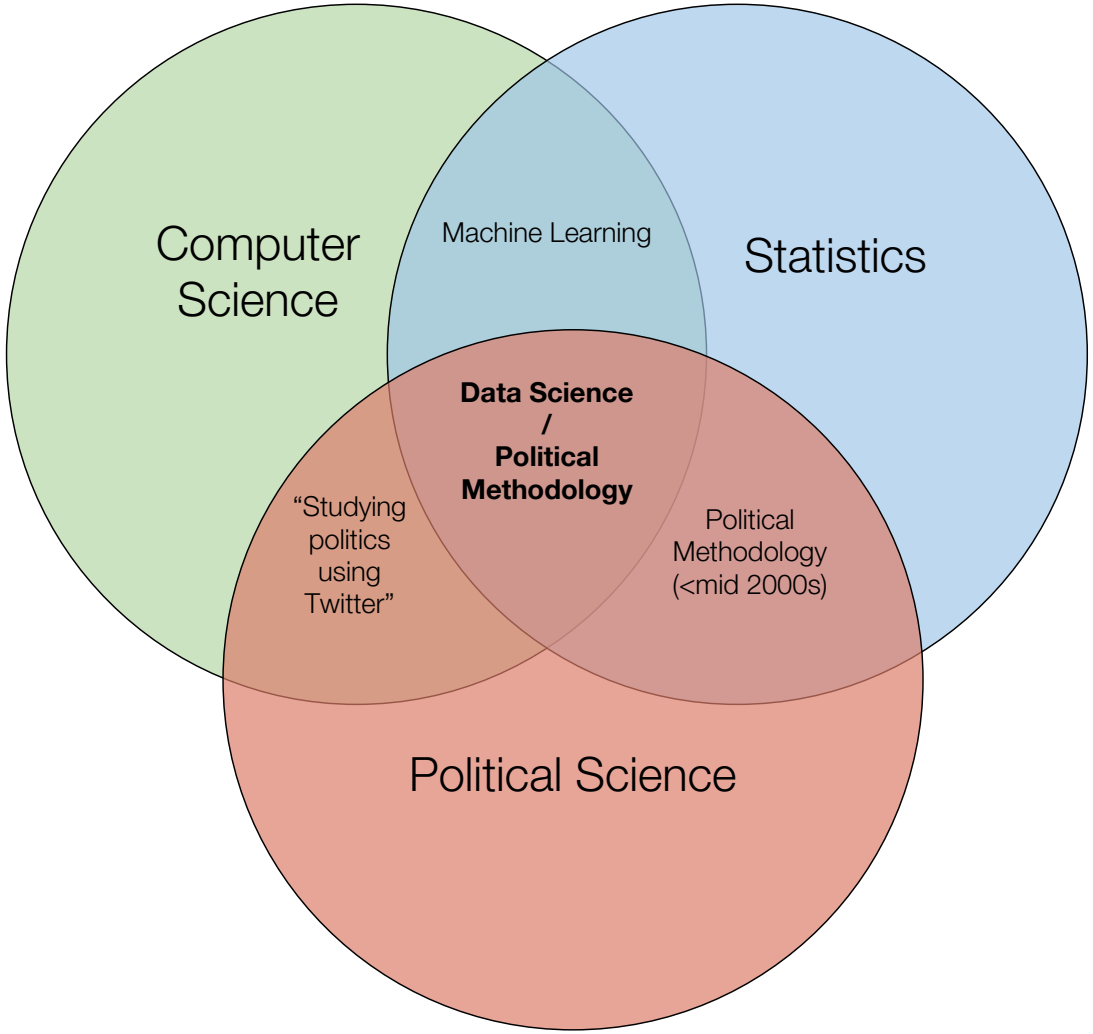
1. Overview [**slides**]
2. Python Fundamentals [**notebook**]
3. Python for Data Science [**notebook**]

Overview

Overview of Data Science



Overview of Data Science



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

Examples in Political Science

- **"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")
```

Examples from https://www.dataquest.io/blog/python-vs-r/

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, shmapply...` 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

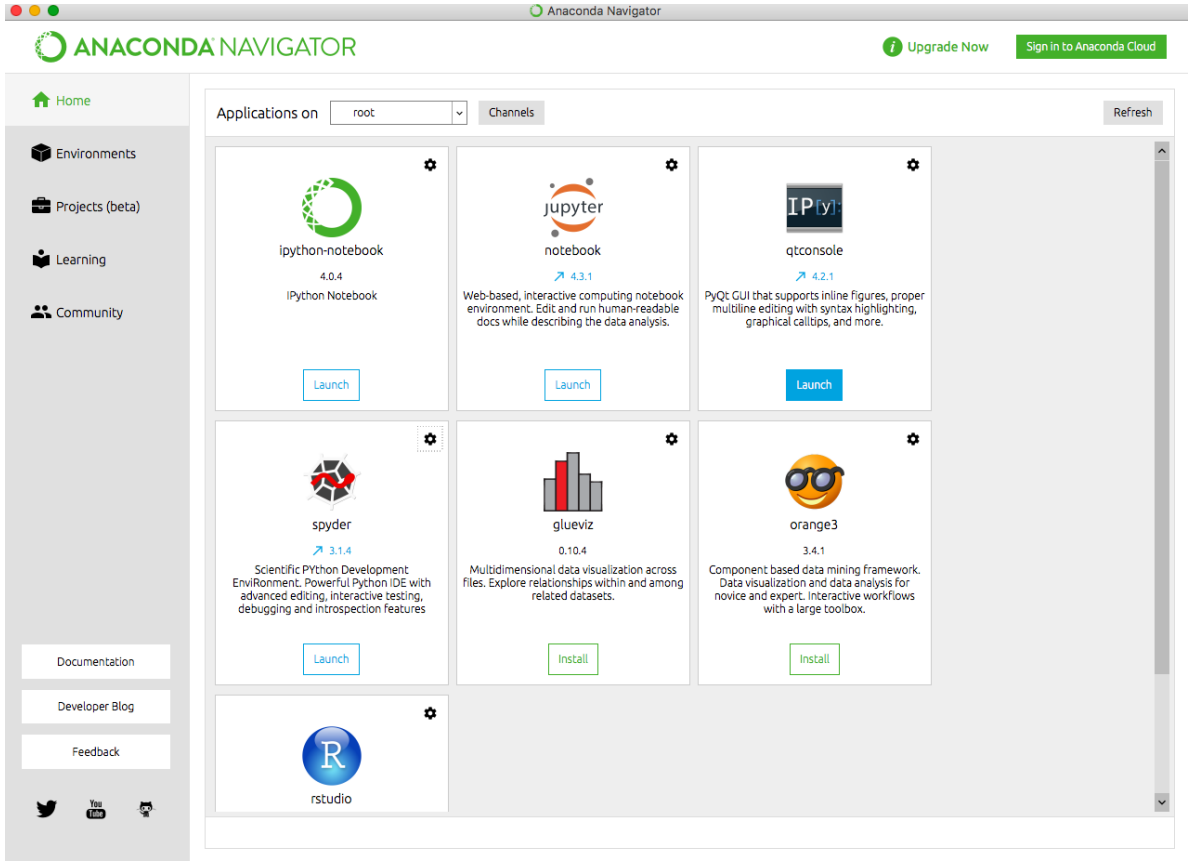
- Building tools vs. Doing analysis
- Flexibility vs. Convenience
- Speed vs. Parallelizability
- 'Computational' vs. 'Statistical'
- Great Machine Learning vs. Okay Machine Learning

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`.

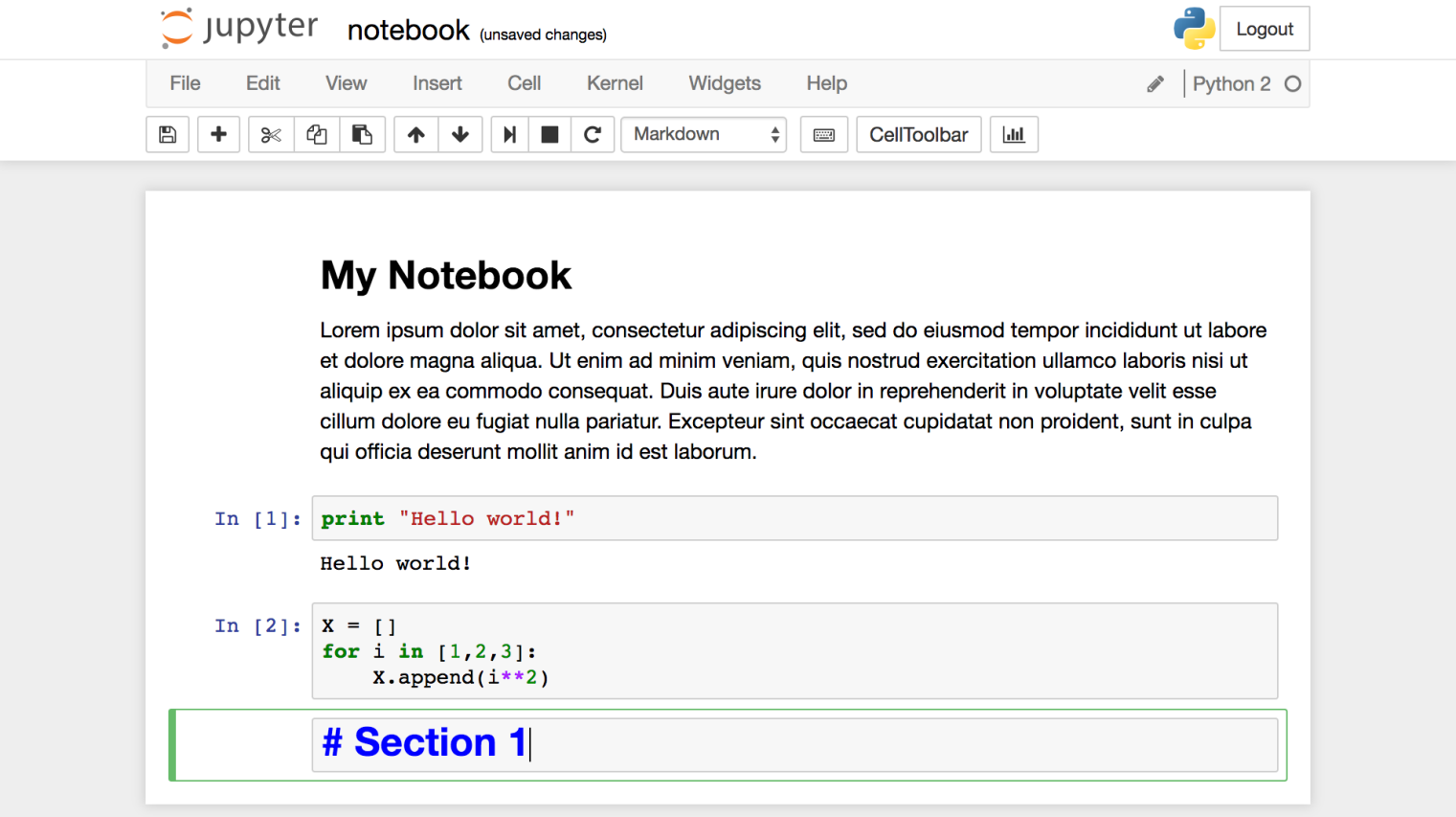
Running Python

A. Command line (**Terminal / Command Prompt**)

```
soubhikbarari — IPython: Users/soubhikbarari — ipython — 119×29
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]:
```

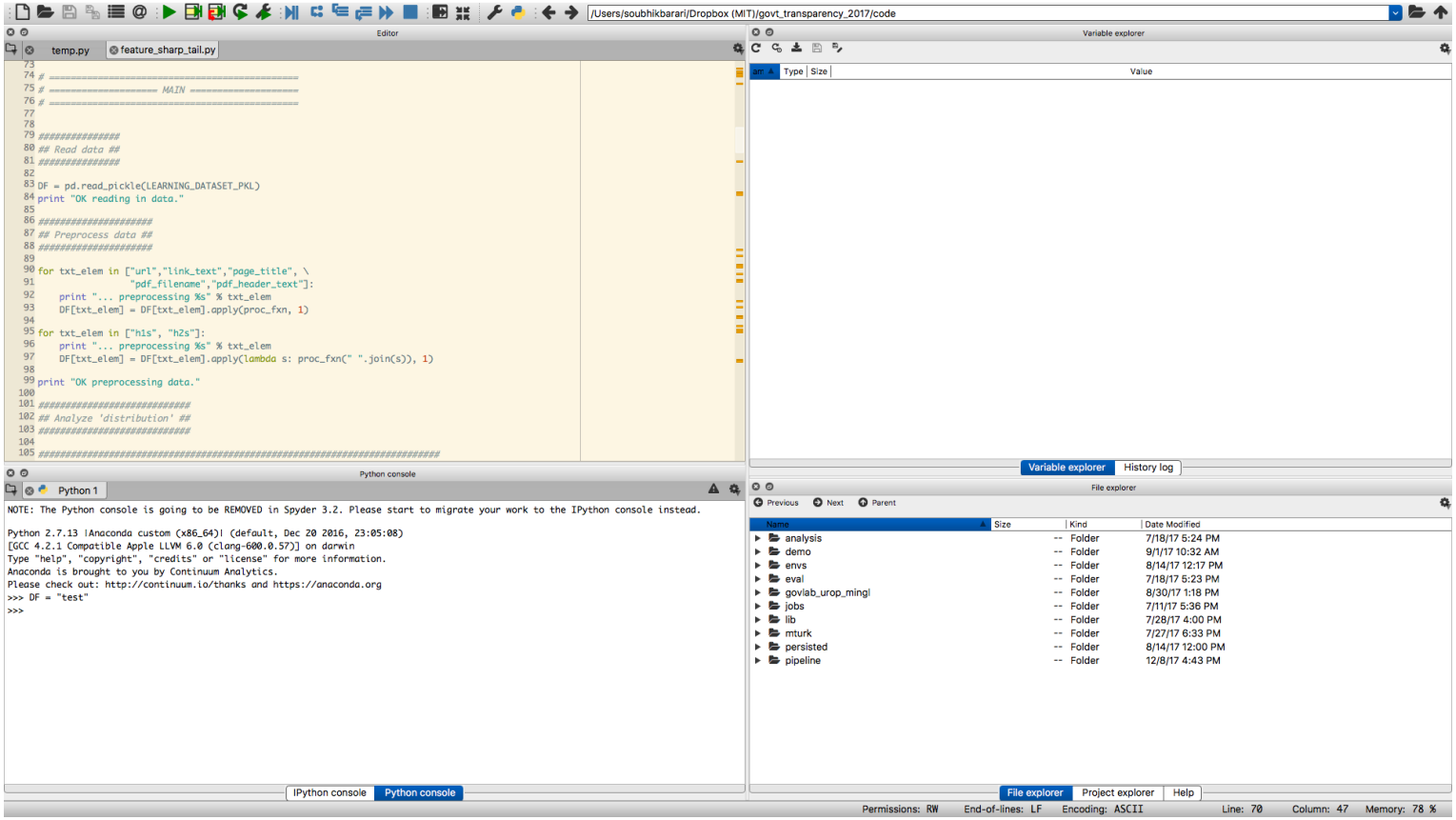
Running Python

B. Notebook (**Jupyter**)



Running Python

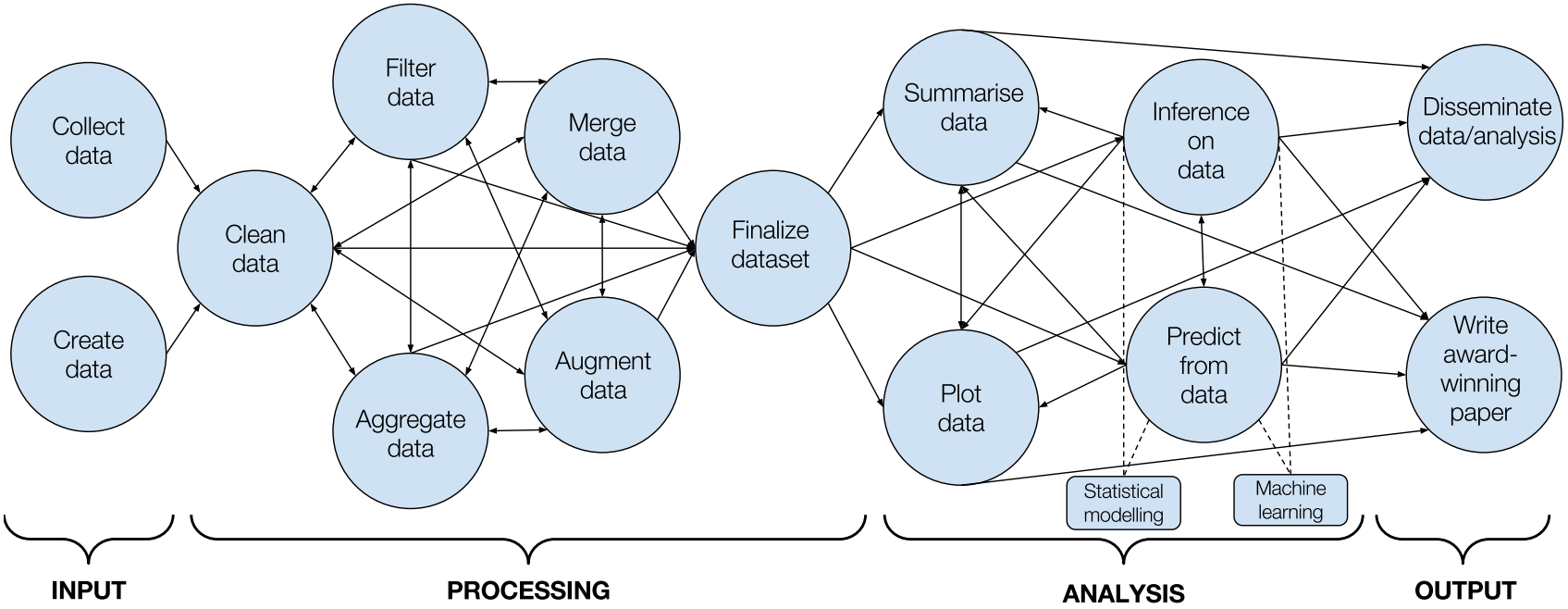
C. IDE (**Spyder**)



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)

Resources

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