

Data Science 1

STAT/CS 287

Jim Bagrow, UVM Dept of Math and Statistics

LECTURE 22

Coding for (data) science

Fundamental challenges

My **goals** for writing scientific code
are, in order of decreasing importance:

1. Correctness
2. Clarity
3. Conciseness
4. Efficiency/speed

High-level and low-level challenges

Coding for (data) science

Fundamental challenges

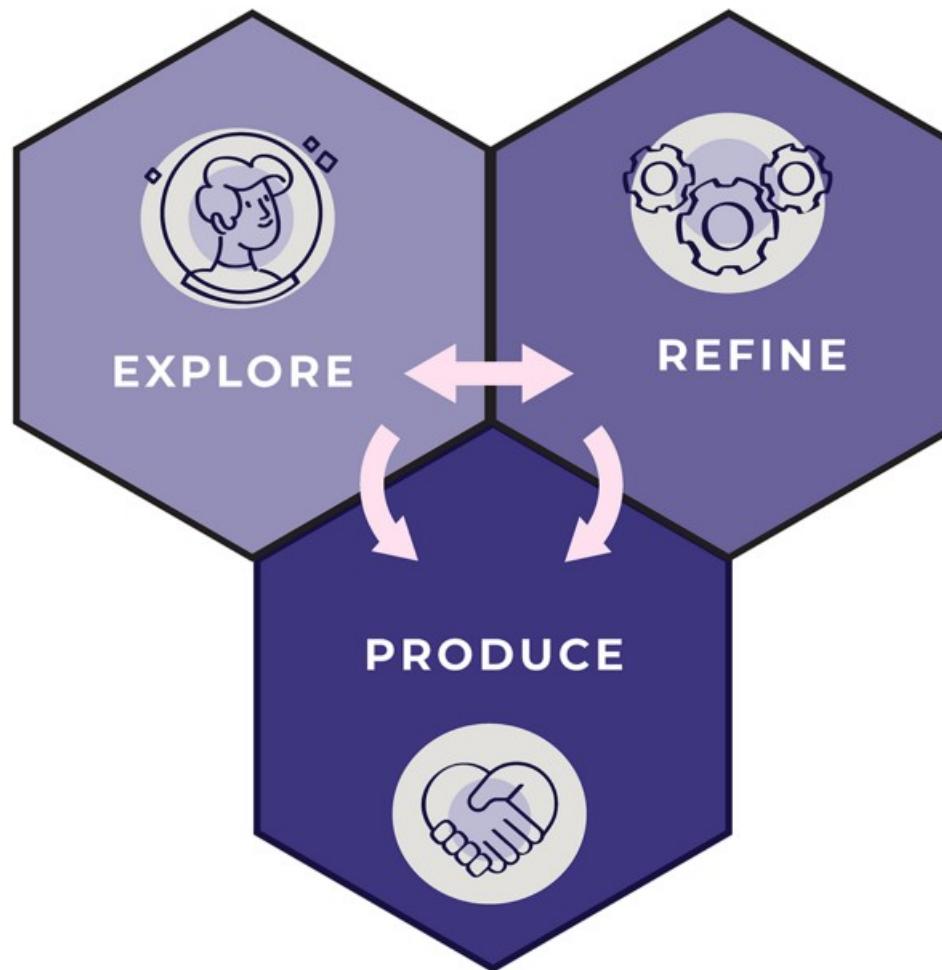
High-level

Low-level

Coding for (data) science

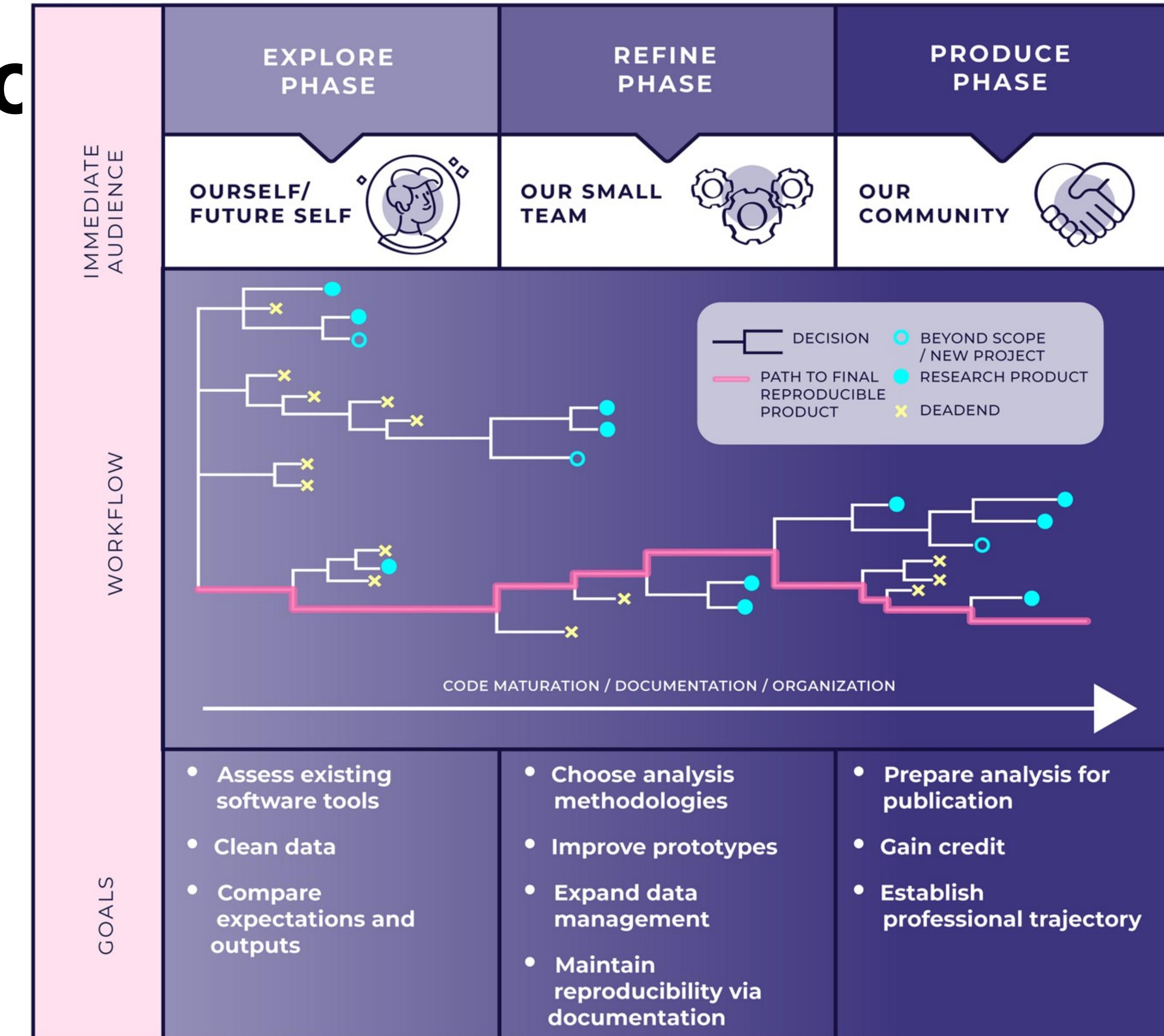
Fundamental challenges

High-level



Low-level

Stoudt, Vásquez, Martinez 2021



Coding for (data) science

Fundamental challenges

High-level

Python 3.6
[\(known limitations\)](#)

```
→ 1 participants = ['Alice',
2                 'Bob',
3                 'Carole']
4
5 part2obs = {}
6 for per in participants:
7     part2obs[per.lower()] = []
```

[Edit this code](#)

Green arrow: line that just executed
Red arrow: next line to execute

0 Step 1 of 11

[Customize visualization](#)

Frames Objects

pythontutor.com

Coding for (data) science

Fundamental challenges

High-level

Python 3.6
[\(known limitations\)](#)

```
→ 1 participants = ['Alice',
2                 'Bob',
3                 'Carole']
4
5 part2obs = {}
6 for per in participants:
7     part2obs[per.lower()] = []
```

[Edit this code](#)

Green arrow: line that just executed
Red arrow: next line to execute

0 Step 1 of 11

[Customize visualization](#)

Frames Objects

pythontutor.com

Some useful references

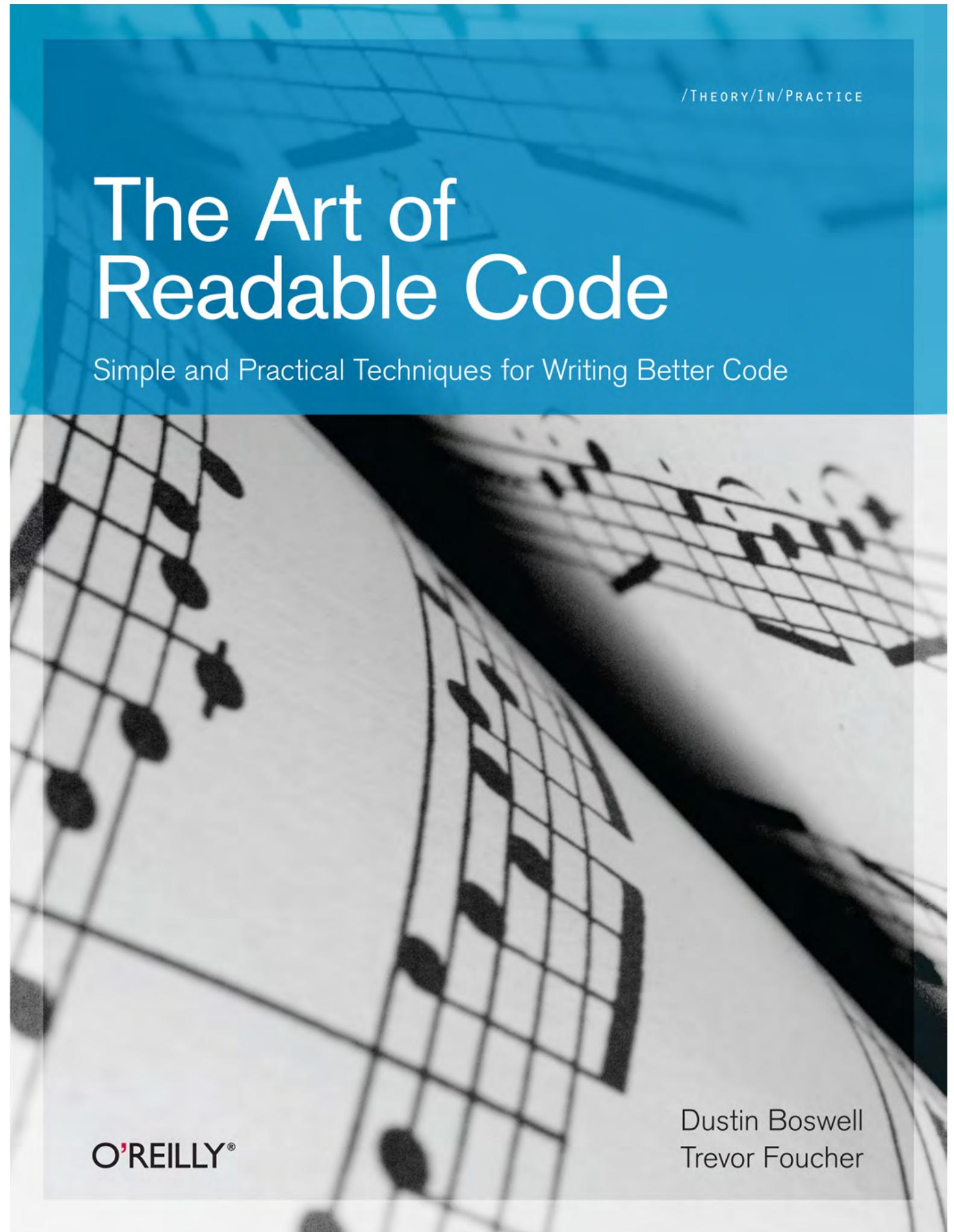


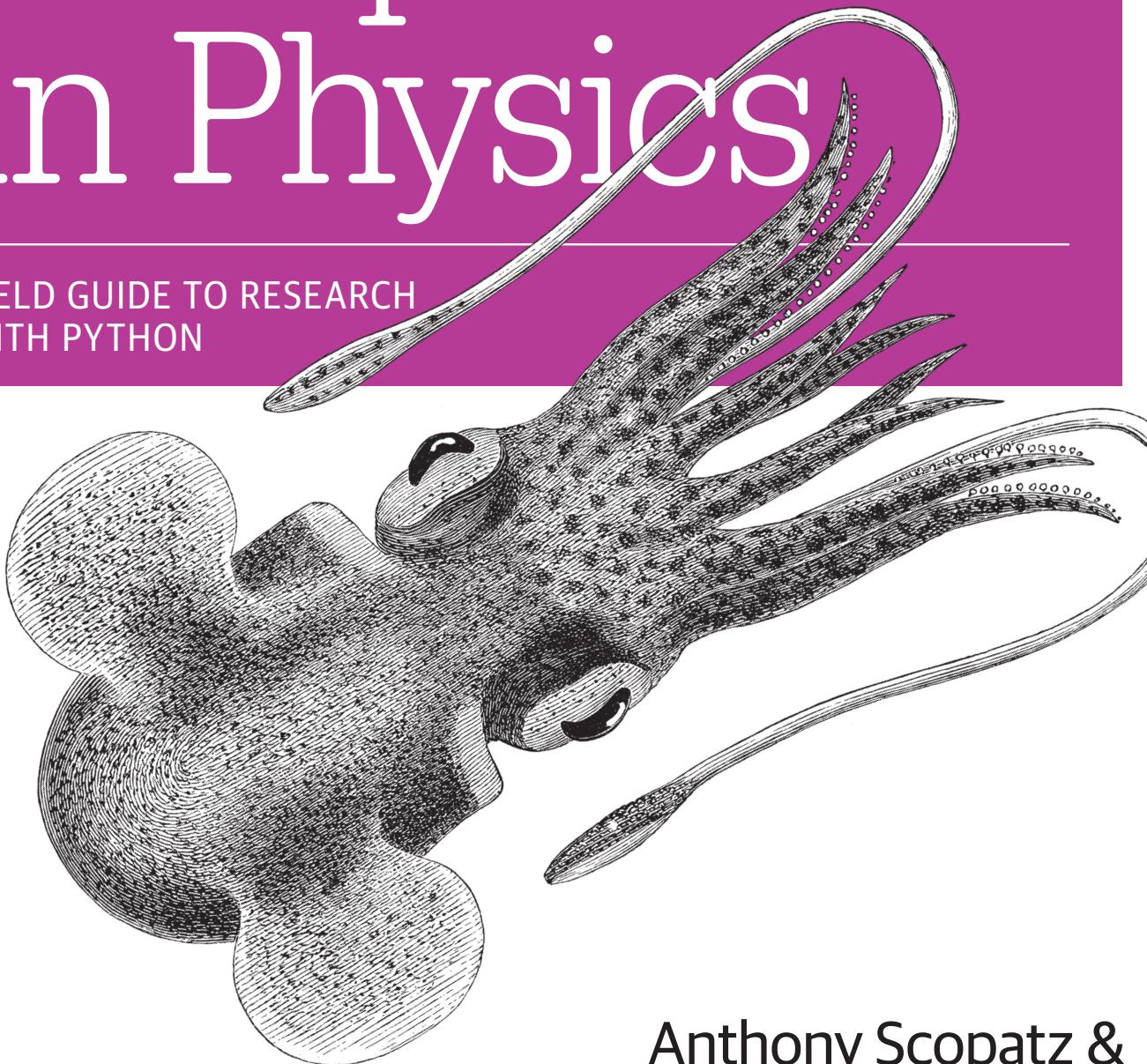
Table of Contents

- › Preface
- › Chapter 1. Code Should Be Easy to Understand
- ▼ Part I. Surface-Level Improvements
 - › Chapter 2. Packing Information into Names
 - › Chapter 3. Names That Can't Be Misconstrued
 - › Chapter 4. Aesthetics
 - › Chapter 5. Knowing What to Comment
 - › Chapter 6. Making Comments Precise and Compact
- ▼ Part II. Simplifying Loops and Logic
 - › Chapter 7. Making Control Flow Easy to Read
 - › Chapter 8. Breaking Down Giant Expressions
 - › Chapter 9. Variables and Readability
- ▼ Part III. Reorganizing Your Code
 - › Chapter 10. Extracting Unrelated Subproblems
 - › Chapter 11. One Task at a Time
 - › Chapter 12. Turning Thoughts into Code
 - › Chapter 13. Writing Less Code
- ▼ Part IV. Selected Topics
 - › Chapter 14. Testing and Readability
 - › Chapter 15. Designing and Implementing a "Minut..."

O'REILLY®

Effective Computation in Physics

FIELD GUIDE TO RESEARCH WITH PYTHON



Anthony Scopatz & Kathryn D. Huff

Table of Contents

- Foreword
- › Preface
- Part I. Getting Started
- › Chapter 1. Introduction to the Command Line
- › Chapter 2. Programming Blastoff with Python
- › Chapter 3. Essential Containers
- › Chapter 4. Flow Control and Logic
- › Chapter 5. Operating with Functions
- › Chapter 6. Classes and Objects
- Part II. Getting It Done
- › Chapter 7. Analysis and Visualization
- › Chapter 8. Regular Expressions
- › Chapter 9. NumPy: Thinking in Arrays
- › Chapter 10. Storing Data: Files and HDF5
- › Chapter 11. Important Data Structures in Physics
- › Chapter 12. Performing in Parallel
- › Chapter 13. Deploying Software
- Part III. Getting It Right
- › Chapter 14. Building Pipelines and Software
- › Chapter 15. Local Version Control
- › Chapter 16. Remote Version Control
- › Chapter 17. Debugging
- › Chapter 18. Testing
- Part IV. Getting It Out There
- › Chapter 19. Documentation
- › Chapter 20. Publication
- › Chapter 21. Collaboration
- › Chapter 22. Licenses, Ownership, and Copyright
- › Chapter 23. Further Musings on Computational Physics

Some useful books

The Art of Readable Code

Simple and Practical Techniques for Writing Maintainable Code

O'REILLY®

- Table of Contents
- > Preface
- > Chapter 1. Code Should Be Easy to Understand
- ▼ Part I. Surface-Level Improvements
 - > Chapter 2. Packing Information into Names
 - > Chapter 3. Names That Can't Be Misconstrued
 - > Chapter 4. Aesthetics
 - > Chapter 5. Knowing What to Comment
 - > Chapter 6. Making Comments Precise and Compact
- ▼ Part II. Simplifying Loops and Logic
 - > Chapter 7. Making Control Flow Easy to Read
 - > Chapter 8. Breaking Down Giant Expressions
 - > Chapter 9. Variables and Readability
- ▼ Part III. Reorganizing Your Code
 - > Chapter 10. Extracting Unrelated Subproblems
 - > Chapter 11. One Task at a Time
 - > Chapter 12. Turning Thoughts into Code
 - > Chapter 13. Writing Less Code
- ▼ Part IV. Selected Topics

O'REILLY®

Effective Computer Programming in Physics

FIELD GUIDE TO RESEARCH WITH PYTHON



James

- > Chapter 1. Introduction to the Command Line
- > Chapter 2. Programming Blastoff with Python
- > Chapter 3. Essential Containers
- > Chapter 4. Flow Control and Logic
- > Chapter 5. Operating with Functions
- > Chapter 6. Classes and Objects
- Part II. Getting It Done
- > Chapter 7. Analysis and Visualization
- > Chapter 8. Regular Expressions
- > Chapter 9. NumPy: Thinking in Arrays
- > Chapter 10. Storing Data: Files and HDF5
- > Chapter 11. Important Data Structures in Physics
- > Chapter 12. Performing in Parallel
- > Chapter 13. Deploying Software
- Part III. Getting It Right
- > Chapter 14. Building Pipelines and Software
- > Chapter 15. Local Version Control
- > Chapter 16. Remote Version Control
- > Chapter 17. Debugging
- > Chapter 18. Testing
- Part IV. Getting It Out There

Compare / Contrast

33 lines, 1583c

James Bagrow · STAT/CS 287 · Data Science 1

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine':  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california':  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        plt.plot(observation[2], observation[3])  
        plt.title('Bedford Falls, New York')  
    elif observation[0][1].lower() == 'derry, maine':  
        plt.plot(observation[2], observation[3])  
        plt.title('Derry, Maine')  
    elif observation[0][1].lower() == 'hill valley, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Hill Valley, California')  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        plt.plot(observation[2], observation[3])  
        plt.title('Stepford, Connecticut')  
    elif observation[0][1].lower() == 'questa verde, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Questa Verde, California')  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        plt.plot(observation[2], observation[3])  
        plt.title('Haddonfield, Illinois')
```

33 lines, 1583c

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine':  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california':  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        plt.plot(observation[2], observation[3])  
        plt.title('Bedford Falls, New York')  
    elif observation[0][1].lower() == 'derry, maine':  
        plt.plot(observation[2], observation[3])  
        plt.title('Derry, Maine')  
    elif observation[0][1].lower() == 'hill valley, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Hill Valley, California')  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        plt.plot(observation[2], observation[3])  
        plt.title('Stepford, Connecticut')  
    elif observation[0][1].lower() == 'questa verde, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Questa Verde, California')  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        plt.plot(observation[2], observation[3])  
        plt.title('Haddonfield, Illinois')
```

values hardwired throughout code

33 lines, 1583c

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york': _____ values hardwired throughout code  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine': _____  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california':  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        plt.plot(observation[2], observation[3])  
        plt.title('Bedford Falls, New York')  
    elif observation[0][1].lower() == 'derry, maine':  
        plt.plot(observation[2], observation[3])  
        plt.title('Derry, Maine')  
    elif observation[0][1].lower() == 'hill valley, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Hill Valley, California')  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        plt.plot(observation[2], observation[3])  
        plt.title('Stepford, Connecticut')  
    elif observation[0][1].lower() == 'questa verde, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Questa Verde, California')  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        plt.plot(observation[2], observation[3])  
        plt.title('Haddonfield, Illinois')
```

33 lines, 1583c

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york': _____ values hardwired throughout code  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine': _____  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california': _____ repetition, redundancy  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        plt.plot(observation[2], observation[3])  
        plt.title('Bedford Falls, New York')  
    elif observation[0][1].lower() == 'derry, maine':  
        plt.plot(observation[2], observation[3])  
        plt.title('Derry, Maine')  
    elif observation[0][1].lower() == 'hill valley, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Hill Valley, California')  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        plt.plot(observation[2], observation[3])  
        plt.title('Stepford, Connecticut')  
    elif observation[0][1].lower() == 'questa verde, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Questa Verde, California')  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        plt.plot(observation[2], observation[3])  
        plt.title('Haddonfield, Illinois')
```

33 lines, 1583c

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york': _____ values hardwired throughout code  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine': _____  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california': _____ repetition, redundancy  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    if observation[0][1].lower() == 'bedford falls, new york': _____ unclear names  
        plt.plot(observation[2], observation[3])  
        plt.title('Bedford Falls, New York')  
    elif observation[0][1].lower() == 'derry, maine':  
        plt.plot(observation[2], observation[3])  
        plt.title('Derry, Maine')  
    elif observation[0][1].lower() == 'hill valley, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Hill Valley, California')  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        plt.plot(observation[2], observation[3])  
        plt.title('Stepford, Connecticut')  
    elif observation[0][1].lower() == 'questa verde, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Questa Verde, California')  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        plt.plot(observation[2], observation[3])  
        plt.title('Haddonfield, Illinois')
```

33 lines, 1583c

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york': _____ values hardwired throughout code  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine': _____  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california': _____ repetition, redundancy  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    if observation[0][1].lower() == 'bedford falls, new york': _____ unclear names  
        plt.plot(observation[2], observation[3])  
        plt.title('Bedford Falls, New York')  
    elif observation[0][1].lower() == 'derry, maine':  
        plt.plot(observation[2], observation[3])  
        plt.title('Derry, Maine')  
    elif observation[0][1].lower() == 'hill valley, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Hill Valley, California')  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        plt.plot(observation[2], observation[3])  
        plt.title('Stepford, Connecticut')  
    elif observation[0][1].lower() == 'questa verde, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Questa Verde, California')  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        plt.plot(observation[2], observation[3])  
        plt.title('Haddonfield, Illinois')
```

33 lines, 1583c

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york': _____ values hardwired throughout code  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine': _____  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california': _____ repetition, redundancy  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    if observation[0][1].lower() == 'bedford falls, new york': _____ unclear names  
        plt.plot(observation[2], observation[3])  
        plt.title('Bedford Falls, New York')  
    elif observation[0][1].lower() == 'derry, maine':  
        plt.plot(observation[2], observation[3])  
        plt.title('Derry, Maine')  
    elif observation[0][1].lower() == 'hill valley, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Hill Valley, California')  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        plt.plot(observation[2], observation[3])  
        plt.title('Stepford, Connecticut')  
    elif observation[0][1].lower() == 'questa verde, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Questa Verde, California')  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        plt.plot(observation[2], observation[3])  
        plt.title('Haddonfield, Illinois')
```

33 lines, 1583c

Compare / Contrast

```
for observation in data:
    if observation[0] == 1:
        list1.append(observation)
    elif observation[0] == 2:
        list2.append(observation)
    elif observation[0] == 3:
        list3.append(observation)
    elif observation[0] == 4:
        list4.append(observation)
    elif observation[0] == 5:
        list4.append(observation)
    elif observation[0] == 6:
        list5.append(observation)

for observation in list1:
    if observation[0] == 1:
        plt.plot(observation[1], observation[2])
        plt.title('Bedi')
    elif observation[0] == 2:
        plt.plot(observation[1], observation[2])
        plt.title('Deri')
    elif observation[0] == 3:
        plt.plot(observation[1], observation[2])
        plt.title('Hilli')
    elif observation[0] == 4:
        plt.plot(observation[1], observation[2])
        plt.title('bedi')
    elif observation[0] == 5:
        plt.plot(observation[1], observation[2])
        plt.title('bedi')
    elif observation[0] == 6:
        plt.plot(observation[1], observation[2])
        plt.title('bedi')

for observation in data:
    if observation[0] == 1:
        +no_water_max5+ne_water_max5+jnt_water_max5+do_water_max5+de_water_max5\
        +dt_water_max5+jno_water_max6+jne_water_max6+jnt_water_max6+fo_water_max6\
        +fe_water_max6+ft_water_max6+mo_water_max6+me_water_max6+mt_water_max6\
        +ao_water_max6+ae_water_max6+at_water_max6+myo_water_max6+mye_water_max6\
        +myt_water_max6+jo_water_max6+je_water_max6+jt_water_max6+jyo_water_max6\
        +jye_water_max6+jyt_water_max6+auo_water_max6+aue_water_max6+aut_water_max6\
        +se_water_max6+st_water_max6+oo_water_max6+oe_water_max6+ot_water_max6\
        +no_water_max6+ne_water_max6+nt_water_max6+do_water_max6+de_water_max6\
        +dt_water_max6+jno_water_max7+jne_water_max7+jnt_water_max7+fo_water_max7\
        +fe_water_max7+ft_water_max7+mo_water_max7+me_water_max7+mt_water_max7\
        +ao_water_max7+ae_water_max7+at_water_max7+myo_water_max7+mye_water_max7\
        +myt_water_max7+jo_water_max7+je_water_max7+jt_water_max7+jyo_water_max7\
        +jye_water_max7+jyt_water_max7+auo_water_max7+aue_water_max7+aut_water_max7\
        +so_water_max7+se_water_max7+st_water_max7+oo_water_max7+oe_water_max7+ot_water_max7\
        +no_water_max7+ne_water_max7+nt_water_max7+do_water_max7+de_water_max7\
        +dt_water_max7+jno_water_max8+jne_water_max8+jnt_water_max8+fo_water_max8\
        +fe_water_max8+ft_water_max8+mo_water_max8+me_water_max8+mt_water_max8\
        +ao_water_max8+ae_water_max8+at_water_max8+myo_water_max8+mye_water_max8\
        +myt_water_max8+jo_water_max8+je_water_max8+jt_water_max8+jyo_water_max8\
        +jye_water_max8+jyt_water_max8+auo_water_max8+aue_water_max8+aut_water_max8\
        +so_water_max8+se_water_max8+st_water_max8+oo_water_max8+oe_water_max8+ot_water_max8\
        +no_water_max8+ne_water_max8+nt_water_max8+do_water_max8+de_water_max8\
        +dt_water_max8+jno_water_max9+jne_water_max9+jnt_water_max9+fo_water_max9\
        +fe_water_max9+ft_water_max9+mo_water_max9+me_water_max9+mt_water_max9\
        +ao_water_max9+ae_water_max9+at_water_max9+myo_water_max9+mye_water_max9\
        +myt_water_max9+jo_water_max9+je_water_max9+jt_water_max9+jyo_water_max9\
        +jye_water_max9+jyt_water_max9+auo_water_max9+aue_water_max9+aut_water_max9\
        +so_water_max9+se_water_max9+st_water_max9+oo_water_max9+oe_water_max9+ot_water_max9\
        +no_water_max9+ne_water_max9+nt_water_max9+do_water_max9+de_water_max9\
        +dt_water_max9+jno_water_max10+jne_water_max10+jnt_water_max10+fo_water_max10\
        +fe_water_max10+ft_water_max10+mo_water_max10+me_water_max10+mt_water_max10\
        +ao_water_max10+ae_water_max10+at_water_max10+myo_water_max10+mye_water_max10\
        +myt_water_max10+jo_water_max10+je_water_max10+jt_water_max10+jyo_water_max10
```

tures

33 lines, 1583c

Compare / Contrast

```
for observation in data:
    if observation[0] == 1:
        list1.append(observation)
    elif observation[0] == 2:
        list2.append(observation)
    elif observation[0] == 3:
        list3.append(observation)
    elif observation[0] == 4:
        list4.append(observation)
    elif observation[0] == 5:
        list4.append(observation)
    elif observation[0] == 6:
        list5.append(observation)

for observation in list1:
    if observation[0] == 1:
        plt.plot(observation[1], observation[2])
        plt.title('Bedi')
    elif observation[0] == 2:
        plt.plot(observation[1], observation[2])
        plt.title('Deri')
    elif observation[0] == 3:
        plt.plot(observation[1], observation[2])
        plt.title('Hilli')
    elif observation[0] == 4:
        plt.plot(observation[1], observation[2])
        plt.title('bedi')
    elif observation[0] == 5:
        plt.plot(observation[1], observation[2])
        plt.title('bedi')
    elif observation[0] == 6:
        plt.plot(observation[1], observation[2])
        plt.title('bedi')

for observation in data:
    if observation[0] == 1:
        +no_water_max5+ne_water_max5+jnt_water_max5+do_water_max5+de_water_max5\
        +dt_water_max5+jno_water_max6+jne_water_max6+jnt_water_max6+fo_water_max6\
        +fe_water_max6+ft_water_max6+mo_water_max6+me_water_max6+mt_water_max6\
        +ao_water_max6+ae_water_max6+at_water_max6+myo_water_max6+mye_water_max6\
        +myt_water_max6+jo_water_max6+je_water_max6+jt_water_max6+jyo_water_max6\
        +jye_water_max6+jyt_water_max6+auo_water_max6+aue_water_max6+aut_water_max6\
        +se_water_max6+st_water_max6+oo_water_max6+oe_water_max6+ot_water_max6\
        +no_water_max6+ne_water_max6+nt_water_max6+do_water_max6+de_water_max6\
        +dt_water_max6+jno_water_max7+jne_water_max7+jnt_water_max7+fo_water_max7\
        +fe_water_max7+ft_water_max7+mo_water_max7+me_water_max7+mt_water_max7\
        +ao_water_max7+ae_water_max7+at_water_max7+myo_water_max7+mye_water_max7\
        +myt_water_max7+jo_water_max7+je_water_max7+jt_water_max7+jyo_water_max7\
        +jye_water_max7+jyt_water_max7+auo_water_max7+aue_water_max7+aut_water_max7\
        +so_water_max7+se_water_max7+st_water_max7+oo_water_max7+oe_water_max7+ot_water_max7\
        +no_water_max7+ne_water_max7+nt_water_max7+do_water_max7+de_water_max7\
        +dt_water_max7+jno_water_max8+jne_water_max8+jnt_water_max8+fo_water_max8\
        +fe_water_max8+ft_water_max8+mo_water_max8+me_water_max8+mt_water_max8\
        +ao_water_max8+ae_water_max8+at_water_max8+myo_water_max8+mye_water_max8\
        +myt_water_max8+jo_water_max8+je_water_max8+jt_water_max8+jyo_water_max8\
        +jye_water_max8+jyt_water_max8+auo_water_max8+aue_water_max8+aut_water_max8\
        +so_water_max8+se_water_max8+st_water_max8+oo_water_max8+oe_water_max8+ot_water_max8\
        +no_water_max8+ne_water_max8+nt_water_max8+do_water_max8+de_water_max8\
        +dt_water_max8+jno_water_max9+jne_water_max9+jnt_water_max9+fo_water_max9\
        +fe_water_max9+ft_water_max9+mo_water_max9+me_water_max9+mt_water_max9\
        +ao_water_max9+ae_water_max9+at_water_max9+myo_water_max9+mye_water_max9\
        +myt_water_max9+jo_water_max9+je_water_max9+jt_water_max9+jyo_water_max9\
        +jye_water_max9+jyt_water_max9+auo_water_max9+aue_water_max9+aut_water_max9\
        +so_water_max9+se_water_max9+st_water_max9+oo_water_max9+oe_water_max9+ot_water_max9\
        +no_water_max9+ne_water_max9+nt_water_max9+do_water_max9+de_water_max9\
        +dt_water_max9+jno_water_max10+jne_water_max10+jnt_water_max10+fo_water_max10\
        +fe_water_max10+ft_water_max10+mo_water_max10+me_water_max10+mt_water_max10\
        +ao_water_max10+ae_water_max10+at_water_max10+myo_water_max10+mye_water_max10\
        +myt_water_max10+jo_water_max10+je_water_max10+jt_water_max10+jyo_water_max10
```

tures

33 lines, 1583c

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine':  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california':  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        plt.plot(observation[2], observation[3])  
        plt.title('Bedford Falls, New York')  
    elif observation[0][1].lower() == 'derry, maine':  
        plt.plot(observation[2], observation[3])  
        plt.title('Derry, Maine')  
    elif observation[0][1].lower() == 'hill valley, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Hill Valley, California')  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        plt.plot(observation[2], observation[3])  
        plt.title('Stepford, Connecticut')  
    elif observation[0][1].lower() == 'questa verde, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Questa Verde, California')  
    elif observation[0][1].lower() == 'haddonfield, illinois':  
        plt.plot(observation[2], observation[3])  
        plt.title('Haddonfield, Illinois')
```

```
locations = ['Bedford Falls, New York',  
            'Derry, Maine',  
            'Hill Valley, California',  
            'Stepford, Connecticut',  
            'Questa Verde, California',  
            'Haddonfield, Illinois']  
  
loc2obs = {}  
for loc in locations:  
    loc2obs[loc.lower()] = []  
  
for observation in dataset:  
    name = observation[0][1].lower()  
    loc2obs[name].append(observation) # keyerror if unexpected loc  
  
for location in locations:  
    obs = loc2obs[location.lower()]  
    times = obs[2]  
    rates = obs[3]  
    plt.plot(times, rates)  
    plt.title(location)
```

21 lines, 577c

33 lines, 1583c

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine':  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california':  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    plt.plot(observation[2], observation[3])  
    plt.title(observation[1])  
  
elif observation[0][1].lower() == 'derry, maine':  
    plt.plot(observation[2], observation[3])  
    plt.title('Derry, Maine')  
  
elif observation[0][1].lower() == 'hill valley, california':  
    plt.plot(observation[2], observation[3])  
    plt.title('Hill Valley, California')  
  
elif observation[0][1].lower() == 'stepford, connecticut':  
    plt.plot(observation[2], observation[3])  
    plt.title('Stepford, Connecticut')  
  
elif observation[0][1].lower() == 'questa verde, california':  
    plt.plot(observation[2], observation[3])  
    plt.title('Questa Verde, California')  
  
elif observation[0][1].lower() == 'bedford falls, new york':  
    plt.plot(observation[2], observation[3])  
    plt.title('bedford falls, new york')  
  
elif observation[0][1].lower() == 'haddonfield, illinois':  
    plt.plot(observation[2], observation[3])  
    plt.title('bedford falls, new york')
```

Good practices

- DRY principle - *don't repeat yourself*
 - lean on functions, data structures (and know how the work)
 - avoid copy-paste
 - let the computer do the work
- isolate data from code
- use informative names
 - treat code as a written document
- Human working memory is limited
 - keep information local
- don't work blind (IPython, etc.)

```
locations = ['Bedford Falls, New York',  
             'Derry, Maine',  
             'Hill Valley, California',  
             'Stepford, Connecticut',  
             'Questa Verde, California',  
             'Haddonfield, Illinois']  
  
loc2obs = {}  
for loc in locations:  
    loc2obs[loc.lower()] = []  
  
for observation in dataset:  
    name = observation[0][1].lower()  
    loc2obs[name].append(observation) # keyerror if unexpected loc  
  
for location in locations:  
    obs = loc2obs[location.lower()]  
    times = obs[2]  
    rates = obs[3]  
    plt.plot(times, rates)  
    plt.title(location)
```

21 lines, 577c

33 lines, 1583c

Compare / Contrast

```
for observation in dataset:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        list1.append(observation)  
    elif observation[0][1].lower() == 'derry, maine':  
        list2.append(observation)  
    elif observation[0][1].lower() == 'hill valley, california':  
        list3.append(observation)  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        list4.append(observation)  
    elif observation[0][1].lower() == 'questa verde, california':  
        list5.append(observation)  
  
for observation in list1+list2+list3+list4+list5:  
    if observation[0][1].lower() == 'bedford falls, new york':  
        plt.plot(observation[2], observation[3])  
        plt.title('Bedford Falls, New York')  
    elif observation[0][1].lower() == 'derry, maine':  
        plt.plot(observation[2], observation[3])  
        plt.title('Derry, Maine')  
    elif observation[0][1].lower() == 'hill valley, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Hill Valley, California')  
    elif observation[0][1].lower() == 'stepford, connecticut':  
        plt.plot(observation[2], observation[3])  
        plt.title('Stepford, Connecticut')  
    elif observation[0][1].lower() == 'questa verde, california':  
        plt.plot(observation[2], observation[3])  
        plt.title('Questa Verde, California')  
  
for observation in dataset:  
    if observation[0][1].lower() == 'haddonfield, illinois':  
        list6.append(observation)  
    elif observation[0][1].lower() == 'bedford falls, new york':  
        list7.append(observation)  
  
for location in locations:  
    toccobs = {}  
    for loc in locations:  
        toccobs[loc] = 0  
    for observation in dataset:  
        name = observation[0][1].lower()  
        if name == location:  
            toccobs[location] += 1  
    times = obs[2]  
    rates = obs[3]/toccobs[location]  
    plt.plot(times, rates)  
    plt.title(location)
```

Good practices

- DRY principle - *don't repeat yourself*
 - lean on functions, data structures (and know how the work)
 - avoid copy-paste
 - let the computer do the work
- **isolate data from code**
- use informative names
 - treat code as a written document
- Human working memory is limited
 - keep information local
- don't work blind (IPython, etc.)

33 lines, 1583c

```
locations = ['Bedford Falls, New York',  
             'Derry, Maine',  
             'Hill Valley, California',  
             'Stepford, Connecticut',  
             'Questa Verde, California',  
             'Haddonfield, Illinois']
```

How to make the code

- work slowly, make incremental changes
- be prepared to revise
- optimize only when needed
- don't write code you don't need
 - code is only useful when it answers your questions

21 lines, 577c

Research template

What about a "pipeline"?

Can you define a rough, reusable
standard organization for (data)
scientific investigations?

Pipeline or workflow:

- [01-load data]
- [02-process data]
- [03-investigate Qs of interest]
- [04-save results]

???

Research template

What about a "pipeline"?

Can you define a rough, reusable
standard organization for (data)
scientific investigations?

Pipeline or workflow:

[01-load data]

[02-process data]

[03-investigate Qs of interest]

[04-save results]

???

```
├── README.md           <- The top-level README for contributors using this project.  
├── .flake8              <- Standard config for Flake 8 linter (run `flake8 src` frequently)  
├── data  
│   ├── 00-external      <- Data from third party sources.  
│   ├── 01-raw            <- The original, immutable data dump.  
│   ├── 02-interim        <- Intermediate data from 00 or 01 that has been transformed.  
│   └── 03-processed       <- The final, canonical data sets  
├── methods-models       <- Trained and serialized models, model predictions, or model summaries  
|                           Other analysis tools archived in separate subdirectories, kept as  
|                           Git submodules, etc.  
├── notebooks            <- Jupyter notebooks. Naming convention is a number (for ordering),  
|                           the creator's initials, and a short '--' delimited description, e.g.  
|                           `01.0-jpb-initial-data-exploration`.  
├── references           <- Data dictionaries, manuals, papers, and all other explanatory materials.  
├── reports  
│   └── figures           <- Generated analysis as HTML, PDF, LaTeX, etc.  
|                           <- Generated graphics and figures to be used in reporting  
├── setup.py              <- makes project pip installable (pip install -e .) so src can be imported  
├── src  
│   ├── __init__.py        <- Source code for use in this project.  
|   │   <- Makes src a Python module  
|   ├── data-gathering     <- Scripts to download or generate data.  
|   |   └── make_dataset.py  
|   ├── analysis-modeling  <- Scripts to train models and then use trained models to make predictions  
|   |   <- (if there is a machine learning component to the project).  
|   |   └── predict_model.py  
|   |   └── train_model.py  
|   ├── data-processing    <- Scripts to process data, for example turn raw data into features for  
|   |   modeling.  
|   |   └── build_features.py  
|   ├── simulations         <- Scripts for performing simulations, if any.  
|   └── visualization       <- Scripts to create exploratory and results oriented visualizations  
|       └── visualize.py  
└── stats                  <- Store intermediary analyses results, data for plotting, etc.
```

Research template

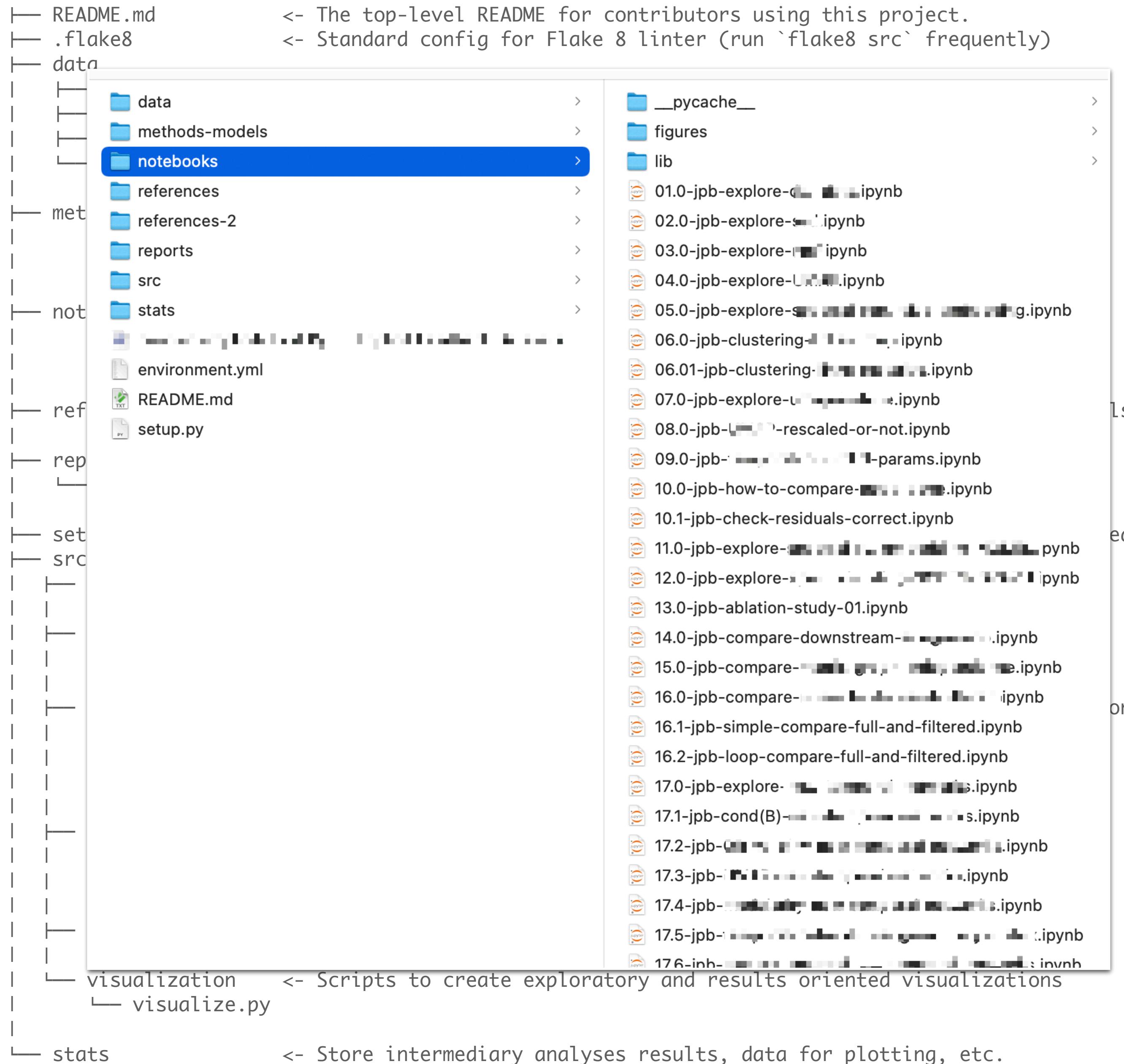
What about a "pipeline"?

Can you define a rough, reusable **standard organization** for (data) scientific investigations?

Pipeline or workflow:

- [01-load data]
- [02-process data]
- [03-investigate Qs of interest]
- [04-save results]

???



Research template

What about a "pipeline"?

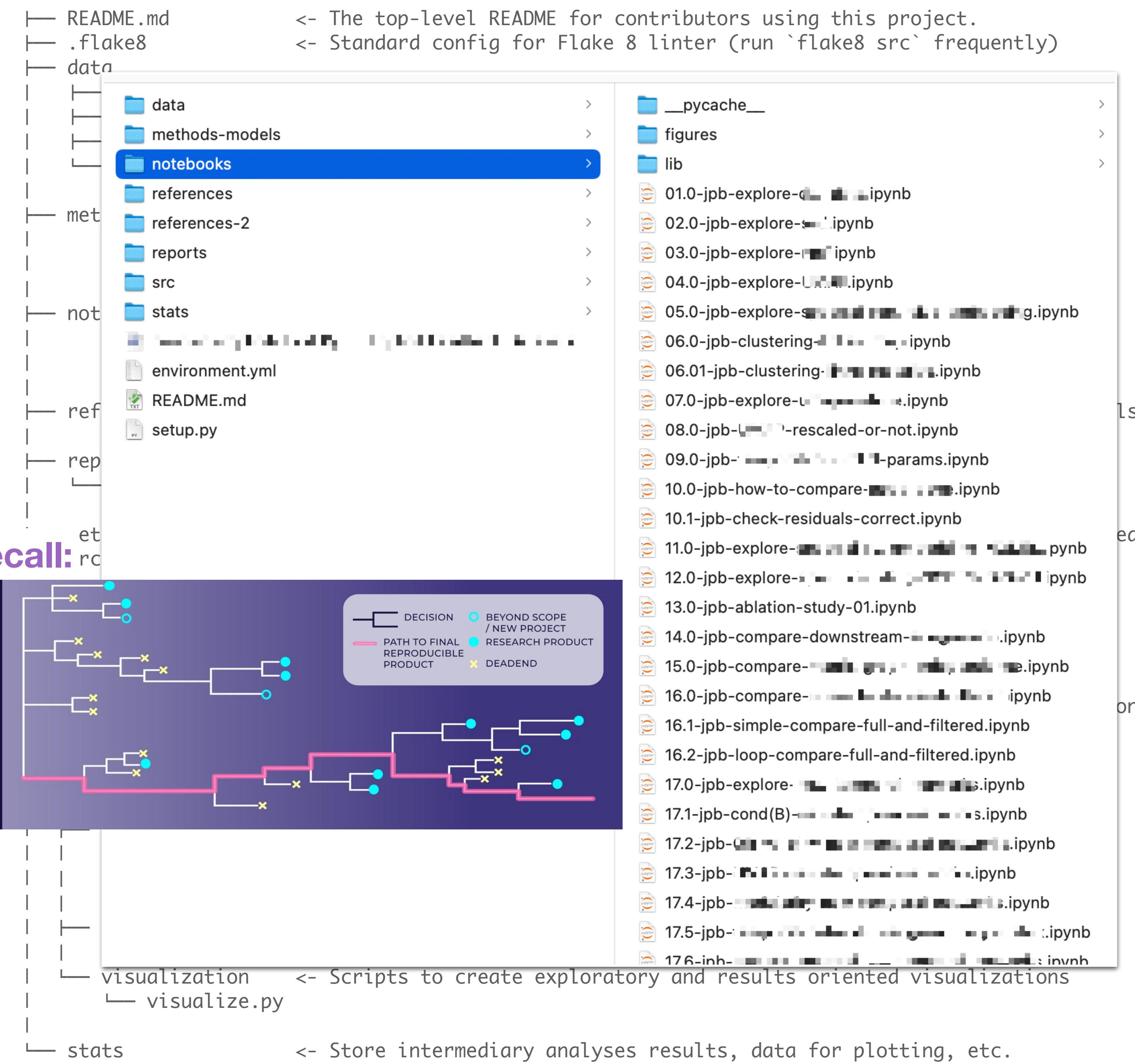
Can you define a rough, reusable **standard organization** for (data) scientific investigations?

Pipeline or workflow:

- [01-load data]
- [02-process data]
- [03-investigate Qs of interest]
- [04-save results]

???

(Can something like this help with the explore-refine-produce loop?)



Team exercise | Coding for (data) science

Team exercise | Coding for (data) science

Plan

1. Break out into your teams
2. Teams choose a team manager
 - keeps discussions on track, makes sure everyone has a turn
3. Each team member in turn:
 - presents their code file(s) for one assignment, what they did and why, what they struggled with, etc.
 - others follow along, taking notes
 - hold a (construction) discussion about the code
 - ~10min per team member, depending on team size and time
4. Any remaining time: work on a research template
 - *Ask me questions at any time*

Team exercise | Coding for (data) science

Plan

1. Break out into your teams
2. Teams choose a team manager
 - keeps discussions on track, makes sure everyone has a turn
3. Each team member in turn:
 - presents their code file(s) for one assignment, what they did and why, what they struggled with, etc.
 - others follow along, taking notes
 - hold a (construction) discussion about the code
 - ~10min per team member, depending on team size and time
4. Any remaining time: work on a research template
 - *Ask me questions at any time*

Deliverables

Team — design a general data science "workflow" template

Individual — Written reflection:

1. My code (~1 page):
 - Could I accomplish as much or more with less?
 - Opportunities for functions, (better) data structures?
 - Defensive data analysis - what will my code do when given bad data? If the data changed?
 - How easily can team members understand my code?
2. Each team member's code (~1 page total):
 - Summary of their work compared to yours
 - How easily can I understand their code
 - What approach did they take to the problem that I can use in the future? That is, what's a takeaway I see from their code?
 - How can they improve?

Team exercise | Coding for (data) science

Plan

1. Break out into your teams
2. Teams choose a team manager
 - keeps discussions on track, makes sure everyone has a turn
3. Each team member in turn:
 - presents their code file(s) for one assignment, what they did and why, what they struggled with, etc.
 - others follow along, taking notes
 - hold a (construction) discussion about the code
 - ~10min per team member, depending on team size and time
4. Any remaining time: work on a research template
 - *Ask me questions at any time*

Deliverables

Team — design a general data science "workflow" template

Individual — Written reflection:

1. My code (~1 page):
 - Could I accomplish as much or more with less?
 - Opportunities for functions, (better) data structures?
 - Defensive data analysis - what will my code do when given bad data? If the data changed?
 - How easily can team members understand my code?
2. Each team member's code (~1 page total):
 - Summary of their work compared to yours
 - How easily can I understand their code
 - What approach did they take to the problem that I can use in the future? That is, what's a takeaway I see from their code?
 - How can they improve?

Logistics

Attendance — taken at end

Files — on Blackboard (coming soon)

Due — last day of classes (see Blackboard)

Grading — bonus:

- Replaces lowest quiz or HW grade, whichever helps more, after lowest grade(s) dropped and only if it helps final grade