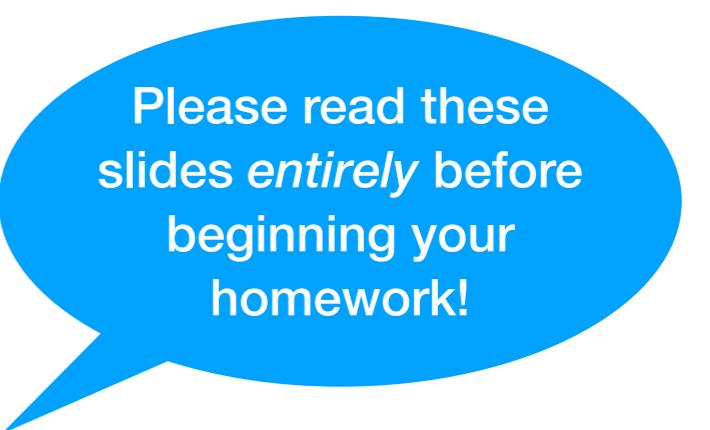


Preparing and submitting assignments

Jim Bagrow
STAT/CS 287 – Data Science 1



Please read these
slides *entirely* before
beginning your
homework!

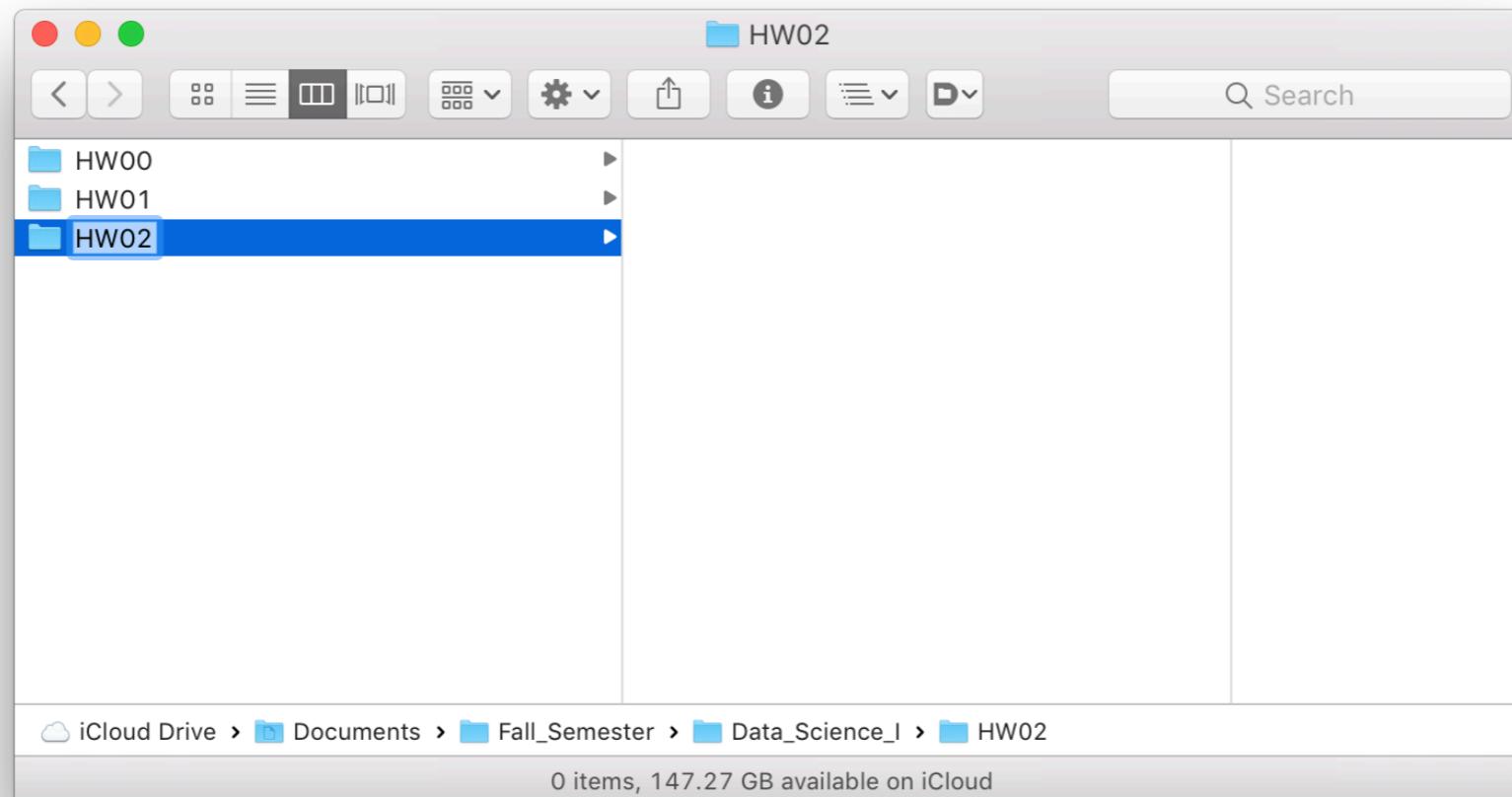
Contents

- Preparing an assignment
 - enclosing folders
 - write-ups
 - including graphics
- Submitting an assignment
 - Submission checklist
- Code specifics:
 - Use relative paths in your code
 - Plots and graphics
 - Using "plotter" scripts
 - Plotter help messages

Preparing an assignment

Preparing an assignment

Create an **enclosing folder** for the assignment (HW or project)

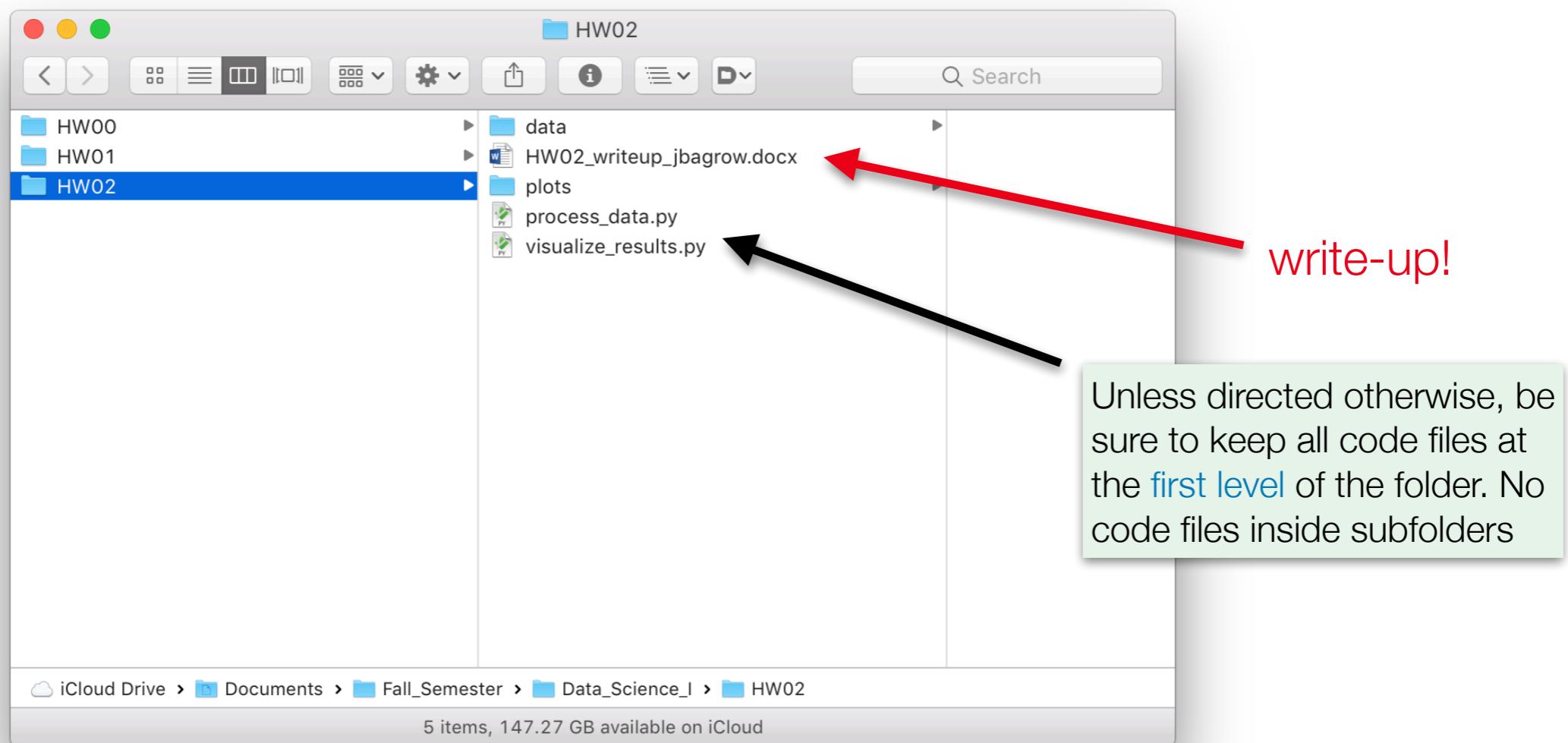


(macOS screenshots for illustration only)

Preparing an assignment

Add files to enclosing folder following assignment directions including **write-up!**

Submissions without a write-up will not be accepted



→ Be sure to name files as instructed

Ex: rename **HW02_writeup_NETID.docx** to
HW02_writeup_jbagrow.docx

Preparing an assignment

Read assignment instructions, perform any requested coding, plotting, etc. and complete the included assignment write-up

Instructions (and problems)

HW 00
STAT/CS 287

Directions

Please read these directions completely before you begin.
Complete the associated HW00_writeup.docx file with your answers.

problem

P1. Do you have Python 3.6 installed using Anaconda Python as per the included instructions?

P2. Have you studied the "Python Setup" slides included with this homework?

P3. Run the `HW00_required_packages.py` script. Does it give any errors or does it finish error-free? If there are errors, install the necessary packages to get it to finish. Now does it run without errors? What packages did you have to install yourself to get it to run without errors?

P4. Please include a screenshot of the **Spyder** editor on your computer showing the `HW00_required_packages.py` file in the editor. (If you have elected not to use Spyder, please instead show screenshot(s) of your setup's *source code editor* and *Python console*.)

P5. The **interactive Python console** (IPython) built into Spyder is powerful, both for running code and for learning about large code bases. To learn about code, IPython provides a **tab completion feature** and the ability to examine code's **docstrings**.

a) Working in IPython, import the numpy package using `import numpy as np` and hitting enter to execute the import statement. Then type `np.r` and immediately hit TAB. You should see a menu of numpy items that begin with the (lowercase) letter "r". Please include a **screenshot** of your IPython console showing this menu.

b) Using IPython, navigate to the numpy "random" sublibrary (`np.random`). Use tab completion—type `np.` (`np` and then a period) and then hit TAB—to see everything that is inside `np.random`. Now, examine the **docstring** of `np.random.shuffle`. Docstrings contain the help and documentation text for different functions inside of Python code.

To see the docstring, add a question mark ("?") after the name of the function and press enter: `np.random.shuffle?` Insert a **screenshot** showing this command and the beginning of the docstring inside your console (you may need to scroll backwards if the docstring is very long).

answer

HW00 — STAT/CS 287
NAME: Jim Bagrow
DATE: 2018-08-27

Remember your name and date!

P1
Here is where to write the answer.

P2

P3

P4

P5a

P5b

P5c

(Write-ups may be prepared outside of Word, but must be formatted exactly as the Word file and *must be submitted as PDF only!* (Word users please submit the Word file.))

Preparing an assignment

Read assignment instructions, perform any requested coding, plotting, etc. and complete the included assignment write-up

Write-up

The screenshot shows a Microsoft Word document window. The title bar reads "HW00_write...". The ribbon menu includes Home, Insert, Design, Layout, References, Mailings, Review, Share, and a search bar. The main content area contains the following text:

HW00 — STAT/CS 287
NAME: Jim Bagrow
DATE: 2018-08-27

P1
Here is where to write the answer.

P2

P3

P4

P5a

Red arrows point from the text "Remember your name and date!" to the "DATE" field and from the text "Please keep these problem headings on their own separate lines in your file" to the "P2" heading.

Remember your name and date!

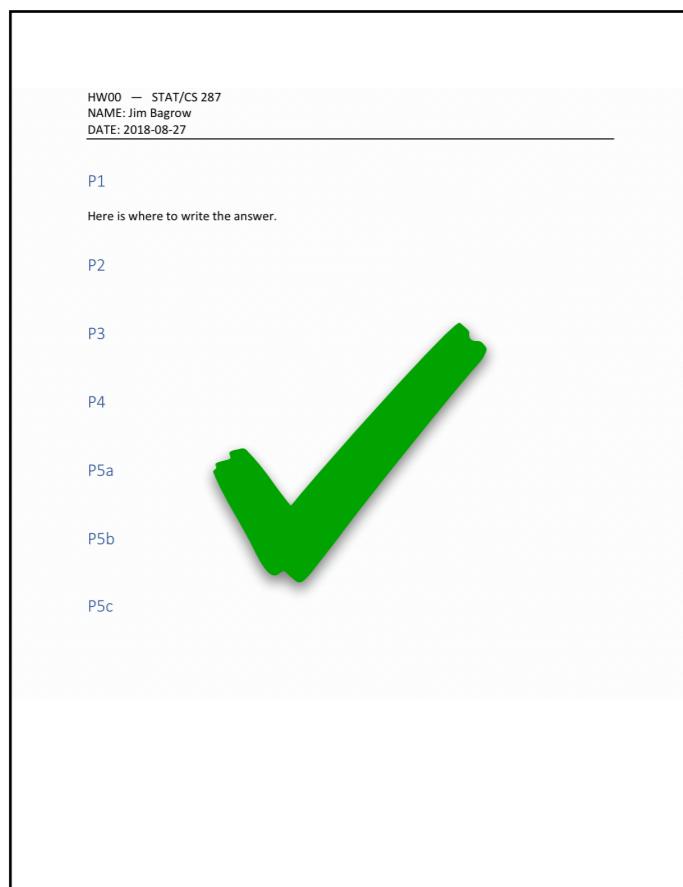
Place written answers, screenshots, plots, etc. after corresponding Px problem heading

Please keep these **problem headings** on their own **separate lines** in your file

Preparing an assignment

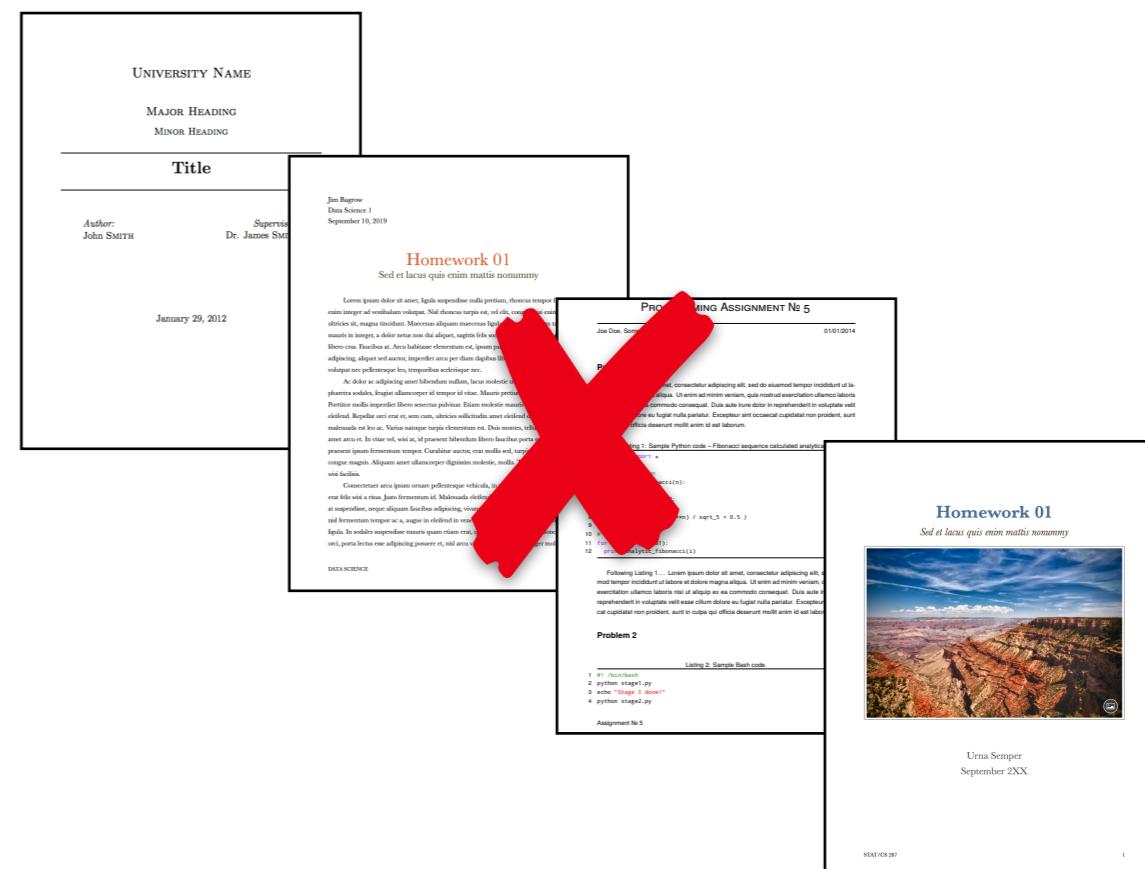
Please follow the write-up format. Do not change fonts, colors, margins, headers, and so forth!

✓ Acceptable



The official write-up template

✗ Unacceptable

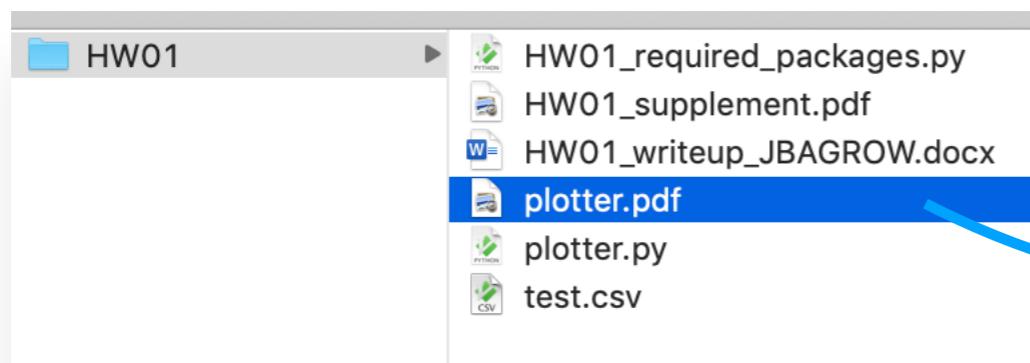


Any other format will be *returned ungraded*

Preparing an assignment

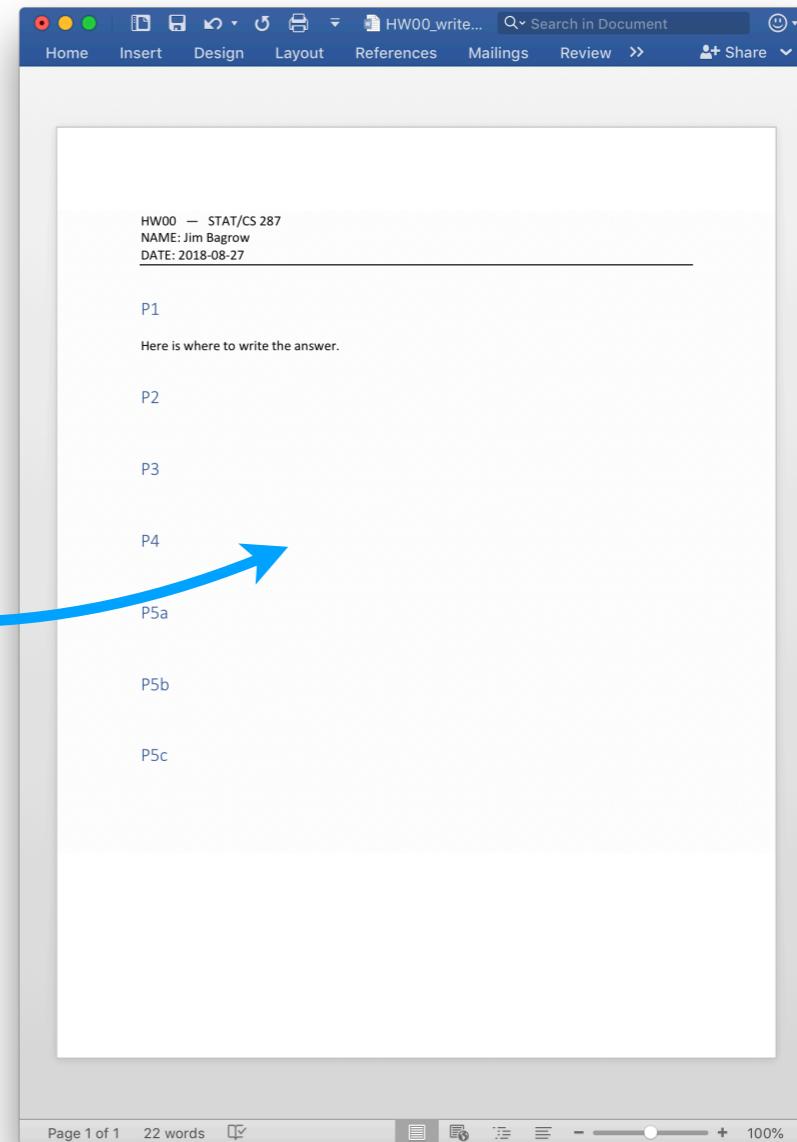
Some problems ask for **graphics**, such as plots or screenshots.

After generating your graphics, you will need to
properly insert them into your write-up



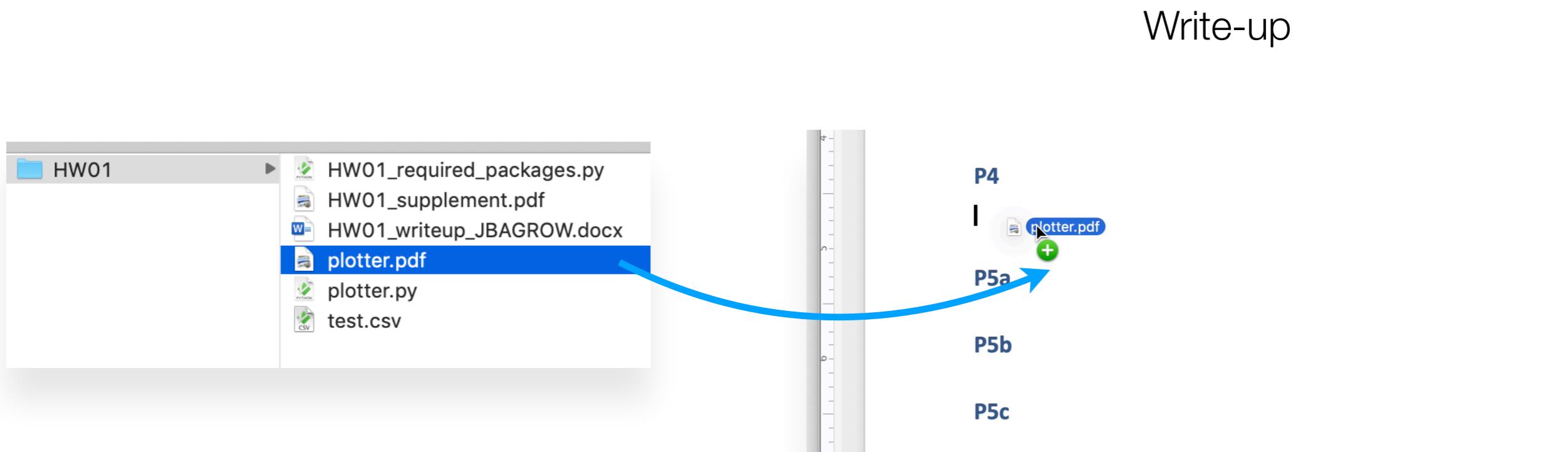
(See "Code specifics" later in this document for details on Python plots)

Write-up



Preparing an assignment

To insert graphics into your write-up

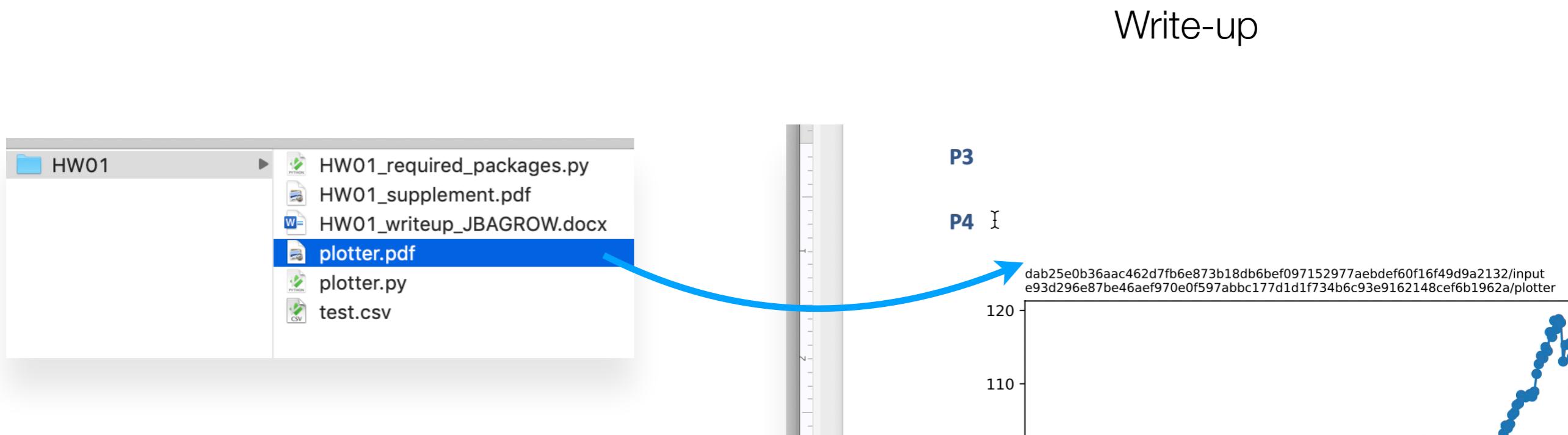


1. Insert figure by **drag-and-drop** or menu command

(Word users on Windows: If the pdf doesn't work in your write-up, try inserting the plotter's .svg file instead. Check the pdf first though!)

Preparing an assignment

To insert graphics into your write-up

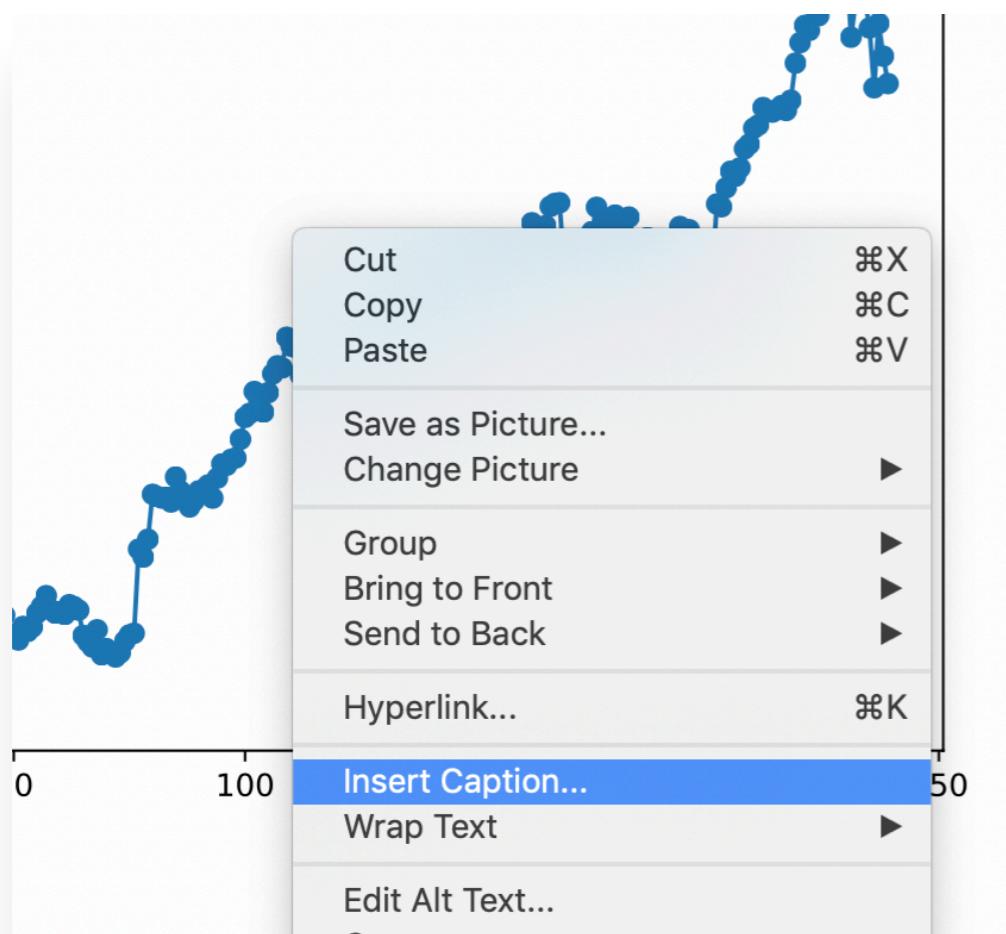


1. Insert figure by drag-and-drop or menu command

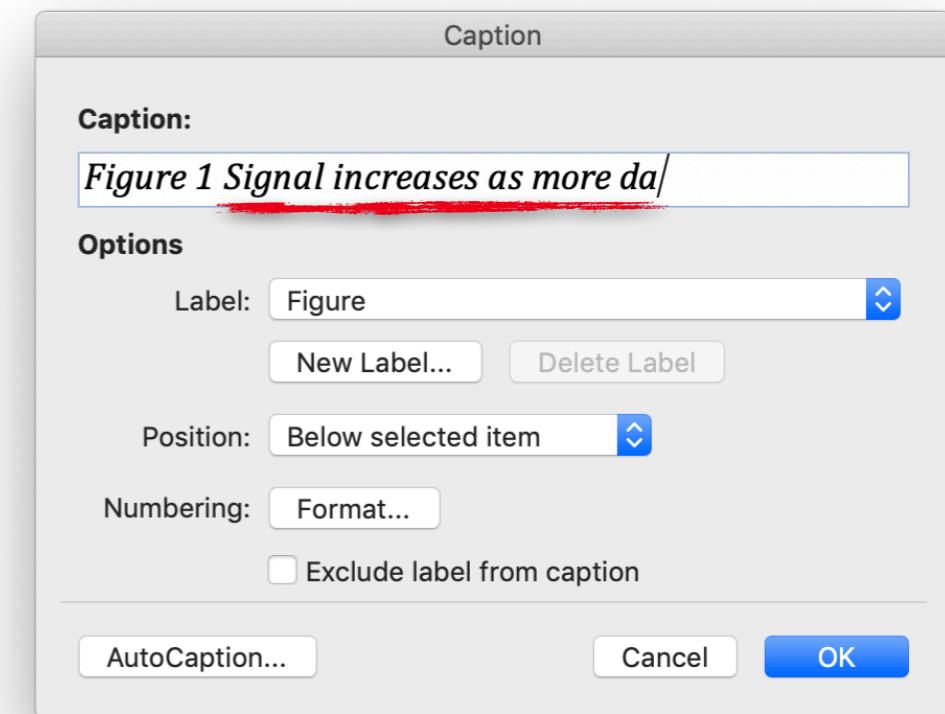
Preparing an assignment

To insert graphics into your write-up

2. Select "Insert Caption...":



3. Type a **descriptive caption**:



Preparing an assignment

To insert graphics into your write-up

4. **Confirm** caption has been added

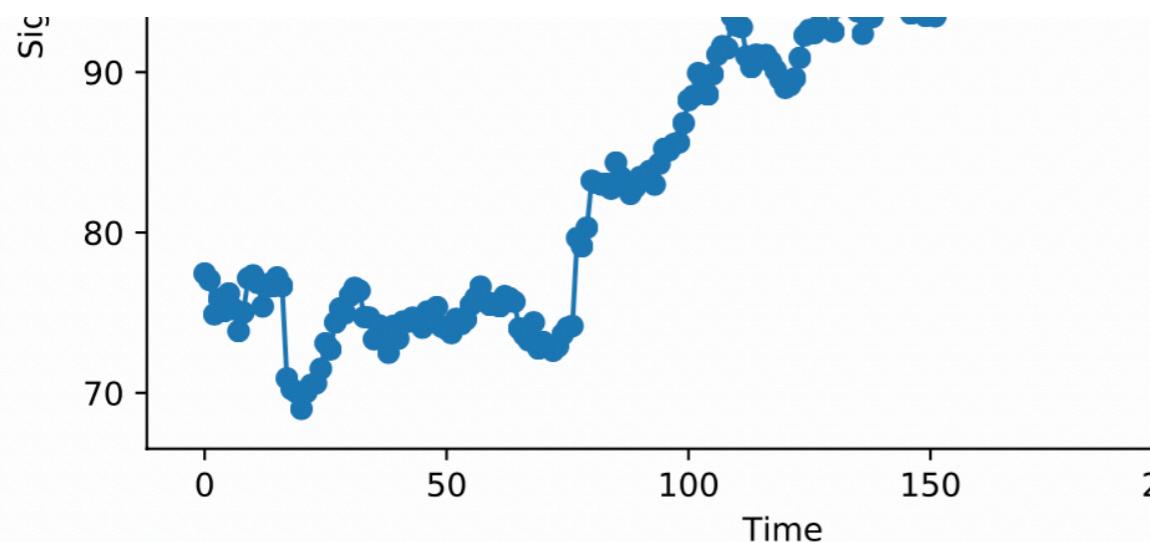


Figure 1 Signal increases as more data are collected.

Taking care to manage the experimental resources we will need

For more information:

- [Office.com: Add, format, or delete captions in Word](#)
- [YouTube: Word 2016 Tutorial Inserting Captions](#)
- [YouTube: Matplotlib figures in MS Word 2016](#)

All captions *must* include descriptive text following the figure number. Writing only "Figure 1", for example, is *not sufficient*.

Preparing an assignment

Graphics Requirements

- All figures must be captioned and placed in appropriate position within write-up
- Ensure figures are numbered appropriately as instructed (Figure 1, Figure 2, etc.)
- Write-ups with unnumbered or uncaptioned figures may be *returned ungraded*

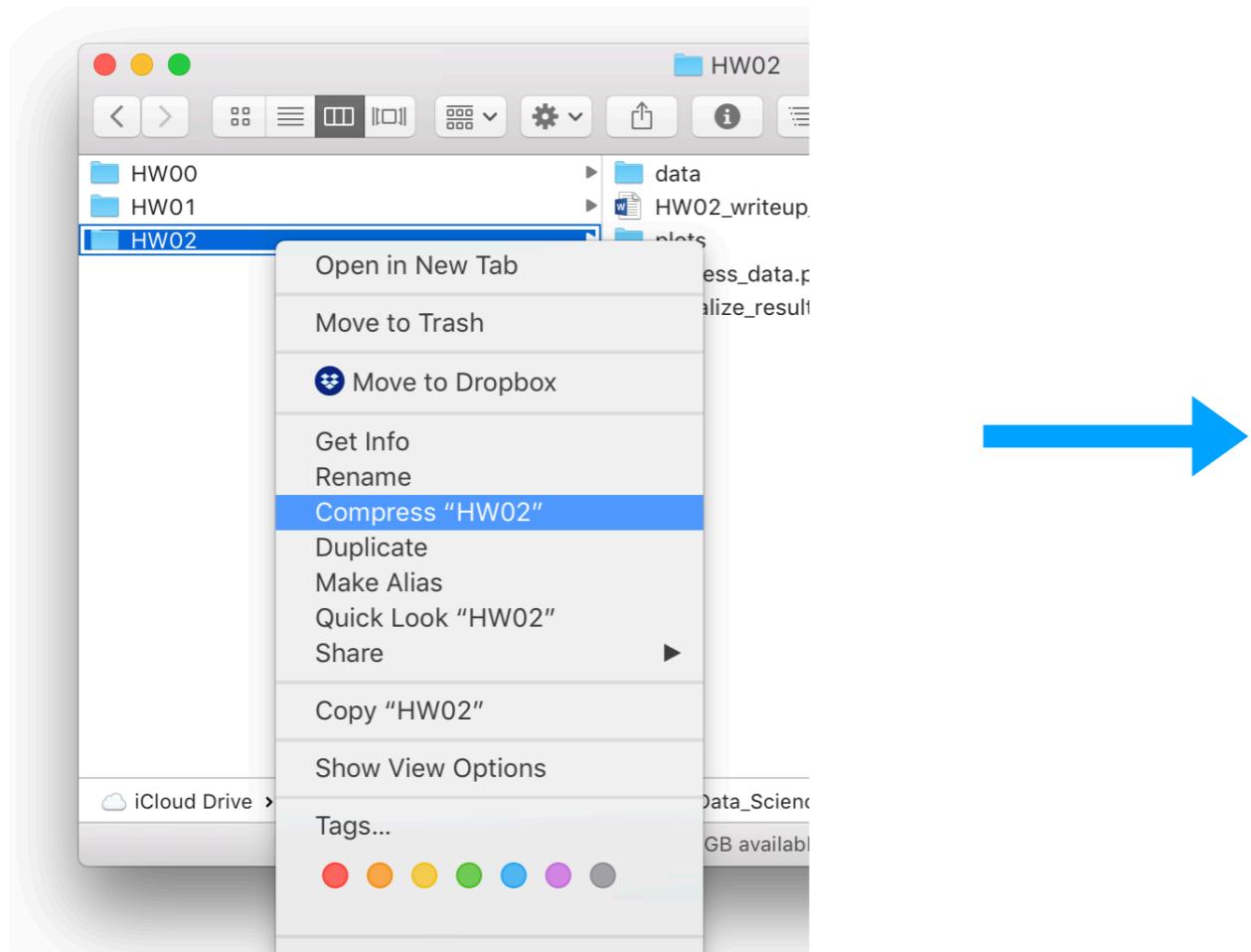
Tables must be captioned just like figures (Table 1, Table 2, etc.)

Always use Word for tables [Office.com: Insert a table](https://Office.com:Insert a table)

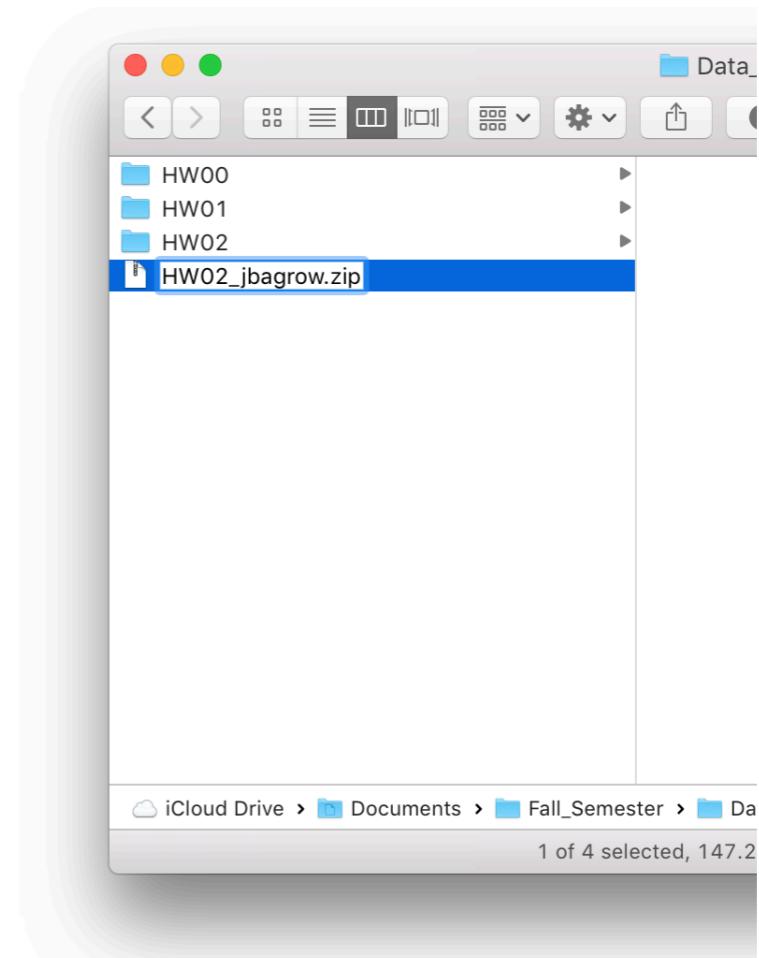
Submitting an assignment

Submitting an assignment

1. Compress enclosing folder to
a *zip** file



2. Rename the zip file according to
assignment instructions as needed



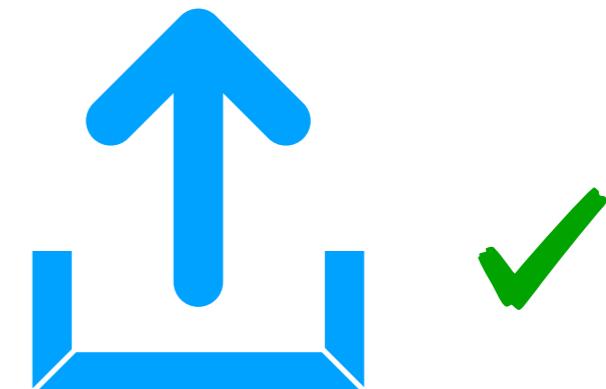
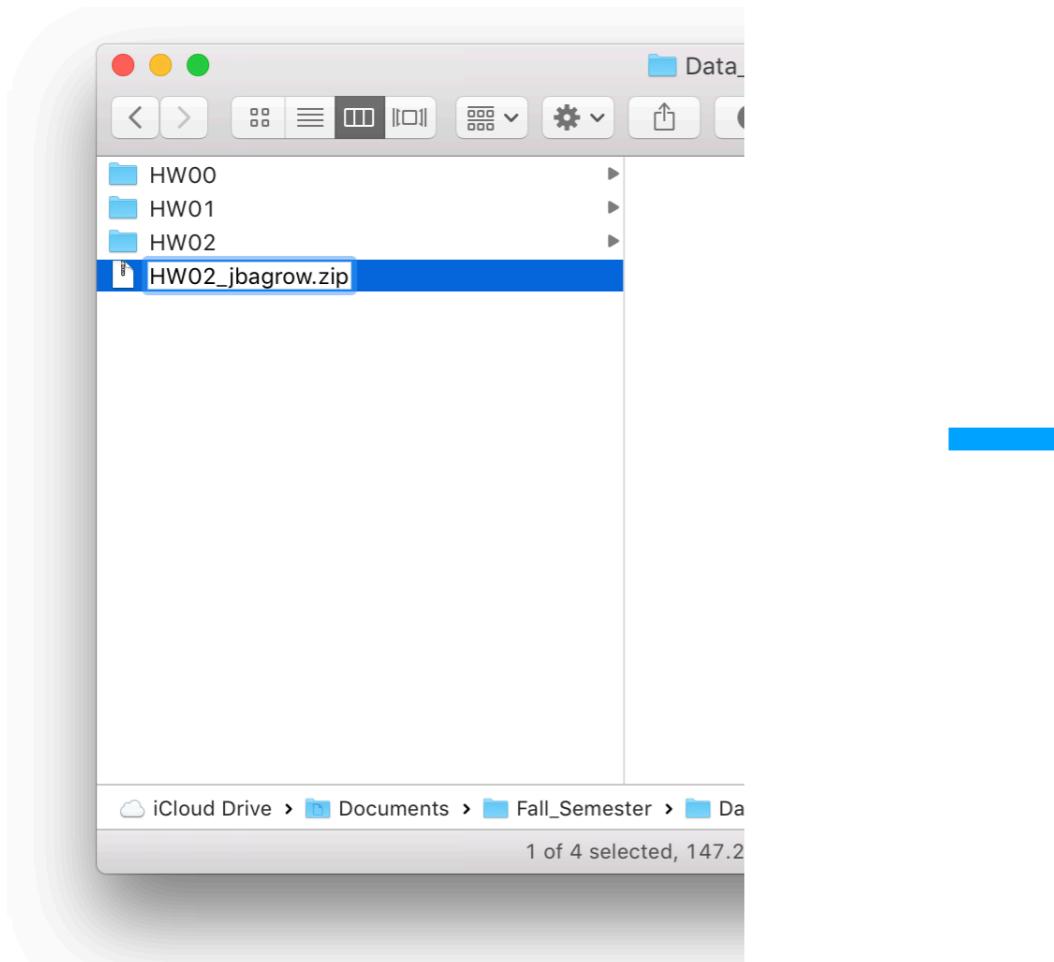
* Use zip, not 7zip, gzip, rar, etc.

Final zip file should be **smaller** than
approx. 50-100 megabytes (you can
exclude large files if necessary)

Submitting an assignment

2. Rename the zip file according to assignment instructions as needed

3. Lastly, [upload](#) to Blackboard



Submission checklist

- Read the entire instructions before you begin the assignment
- File and folder names match assignment instructions
 - Any [plotter scripts](#) have been used appropriately
- Python code included
 - [No absolute paths](#) in your Python code
- Write-up complete
 - Your correct name and date are inside the write-up
 - Plots and graphics appropriately formatted and labeled (captioned)
- Enclosing folder compressed to a zip file and uploaded to **Blackboard**
 - Large files excluded** if necessary; final upload smaller than 50-100 megabytes

Code specifics

Use *relative* paths in your code

Stay **inside the enclosing folder** to make assignments ***self-contained***

Make sure any **code** you submit only reads files from locations within the enclosing folder and only writes files to locations within the enclosing folder

Use **relative paths** not **absolute paths**



```
open("data/experiment1/example.csv")
pyplot.savefig("plots/prev_day_counts.pdf")
```

relative to
enclosing folder



```
open("C:\user\docs\DS1\HW02\statistics\output01.txt", 'w')
pyplot.savefig("/Users/jbagrow/Documents/Fall_Semester/Data_Science_I/HW02/plots/
prev_day_counts.pdf")
```

this code won't run on
another computer!

Paths should *not contain references* to
locations outside the enclosing folder

- Be sure to name any files as instructed

See course reading for more on *file paths*

See also: *working directories*

Plots and graphics

As discussed above ([Preparing an assignment](#)), you will be asked to generate **graphics**—such as screenshots or plots—for this course

- **Screenshots** can be captured using tools built into your computer operating system, and are typically generated as png or jpeg (raster) images.
- **Plots** will usually be made using [Matplotlib](#), a popular third-party Python library

Many assignments, especially early on, will come with **plotter scripts**. You will need to *prepare the files read by these scripts*, run the scripts, then [insert the created plots into your write-up](#).

The first homework comes with an example plotter script and input (data) file, to learn how these work.

Eventually, you will need to generate your own plotting code

Using plotter scripts

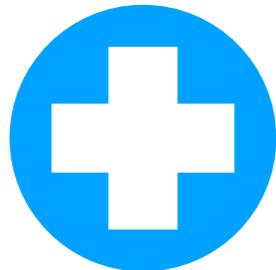
HW01 comes with a [simple plotter script](#) (`plotter.py`) for you to become acclimated to using it. This script reads a provided text file (`test.csv`) and generates a simple plot (`plotter.pdf`).

- Future assignments may contain more complex plotter scripts and you will need to generate the input file(s)
- Eventually, you will need to make plots without being given a script
- All plots must be inserted into your write-ups according to the requirements given above ("[Preparing an assignment](#)")



Unless instructed otherwise, **do not modify any plotter scripts**:

1. Hashes are embedded in each plot file that can detect if the plotter script has been changed.
2. If the script is modified, or not used at all, the assignment may be returned **ungraded**.



All plotter scripts come with **descriptive help** messages to learn what input(s) they require



Plotter help messages

Plotter scripts are run like any Python script (see Python setup / Running code)

Plotter scripts can also **print a help message** instead of running. Here's how to do this in Spyder:

1. Running the script by pressing the "play" button
 - 1a. Plot is saved in this case to `plotter.pdf` → goes into **write-up**
2. Manually entering "`run plotter.py -h`" or "`run plotter.py --help`" into the console will instead display the help message
 - 2a. Here the help message tells us that a file is read called "`test.csv`"

IPython console inside Spyder:

1
In [1]: `runfile('/Users/bagrowjp/teaching/STAT287_F2019/HW01/plotter.py', wdir='/Users/bagrowjp/teaching/STAT287_F2019/HW01')`
File created: plotter.pdf at location /Users/bagrowjp/teaching/STAT287_F2019/HW01

dab25e0b36aac462d7fb6e873b1 097152977aebdef60f16f49d9a2132/input
e93d296e87be46aef970e0f597abbcc1/d1d1f734b6c93e9162148cef6b1962a/plotter

1a

2
In [2]: `run plotter.py -h`
usage: `plotter.py [-h]`

This is a TEST PLOTTER SCRIPT for your first homework. In the future, **2a** will need to prepare the input file(s) (in this case, the input (test.csv) already been provided). For plotter scripts where you generate the input file, use the help string to understand the format of input file. Assignments with figures generated outside plotter scripts when a plotter script should have been used will NOT BE ACCEPTED. Likewise, assignments using plotter scripts you have modified will not be accepted.

optional arguments:
 `-h, --help` show this help message and exit