Creating Interactive Computational Learning Experiences with Jupyter

Jonathan Graves

2023-05-30

Table of contents

0.1 About These Notes

> ⚠ Important
>
> Please read this preamble carefully!

These notes are intended to help you follow the presentation in this workshop.

- Most of the workshop will be **hands-on** in which I will demonstrate the material and techniques; you should code along!
- In Section 3, I provide a **quick-reference** guide to the different code functions we will use
  - This will accompany the presentation, in case you get lost or want to look up something
- In Section 5, I provide a collection of supplemental readings and tips for those interested in more content.

Because of this format, these notes are likely *not* standalone: if they're not completely clear, that's OK. I'll go through it in the presentation.

1 Introduction

1.1 Pitch: A Common Situation

Have you ever:

- Wanted to show students some cool patterns in data?
- Needed to teach students how to do basic coding?
- Tried to demonstrate a new technique or something?

This is a *major* learning context in any course which deals with or teaches data and computation.

1.2 Problem

However, this is actually way harder to do than it should be:

- Your students need computers that can run the material…
- They need to have the right software…
- They need to know how to use it - maybe code it?
- You need to share it with them

All of these are major points of failure.

1.3 A Solution: Jupyter

Figure 1: Jupyter to the rescue

- The Jupyter Project

1.4 Big Picture

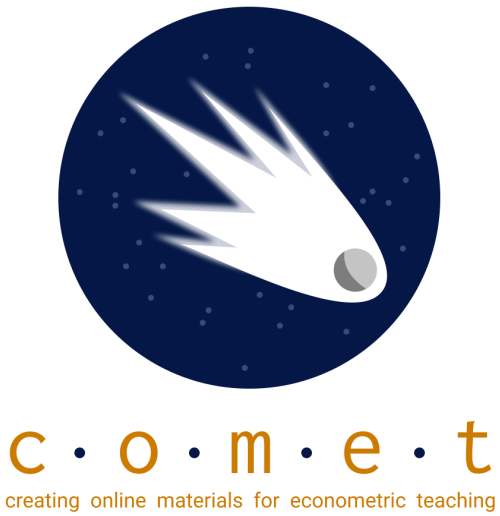

Figure 2: COMET Project Logo

COMET Notebook Example

1.5 Just One Option

There are many other ways to use them

- As demonstrations
- As tests or projects
- As websites

Etc, etc, etc.

1.6 This Workshop

In this 90 minute workshop, I'm going to give you a hands-on tour of the Jupyter environment, and show you how you can create these kinds of experiences.

- Introduction to Jupyter at UBC

- Basic Notebooks and Options
- Making Your First Notebook
  - Writing Markdown Code
  - Writing Executable Code
  - Embeds and Images
- Assessment and Advanced Use
- Q&A

1.7 Learning Objectives

By the end of this workshop, you will be able to:

- Create a basic Jupyter notebooks on UBC's JupyterOpen
- Add markdown cells with a mixture of different content
- Create basic code cells and run them
- Understand interactivity and self-testing
- Write a basic self-test
- Understand different deployment options

2 Hands-On

2.1 Get Started

We will do this using the following website: UBC JupyterOpen. There are two models:

1. JupyterOpen: a general-purpose hub
2. JupyterCourse: a course-specific hub

In general, (1) is more suitable unless you have need (a) *large* amounts of data, (b) very specific technical requirements, (c) Canvas assessment integration.

- Non-UBC general hub: https://syzygy.ca/
- Back-up hub: https://jupyter.org/try-jupyter/lab/index.html

2.2 Working with the Project!

At this point, we're going to do this as a live-coding exercise.

- Follow along!
- If you get stuck, raise your hand
- Ask questions at any point!

Check out the Quick-Reference section for some specifics.

2.3 Goals

0. Introduce JupyterLab
1. Make a new notebooks and rename it
2. Add a markdown title cell
3. Add some markdown text
    1. Bullets, text-decorations, code, math
4. Embed a video
5. Create a code cell and run it
6. Create a self-test
7. Encapsulate self-tests

3 Quick-Reference

3.1 Markdown

- # are headings; H1 = #, H2 = ## etc.
- * around text is italics, ** is bold, *** is bold italics
- > at the start of a line is a blockquote
- * at the start of a line followed by a space is a bullet point; two spaces indents the list
- ` are code literals: use three for fenced code

3.2 Markdown Examples

```
# My Heading

* Bullet One
* Bullet Two
  * Indented Bullet
```

```
*Italics* and **bold** and ***more***

> A quote

`Some code`

```
In a block
```
```

3.3 Links and Images

- Basic link is enclosed in <>: `<www.google.ca>`
- URL is `[text](link)`: 'Google Page
- Images are: `[caption](path/URL)`: `[My cat](media/cat.jpg)`

3.4 Math

Math is standard LaTeX/MathJaX format: `$` or `$$` for display mode.

- `$y = mx + b$` is inline
- For display mode:

```
$$
y = mx + b
$$
```

3.5 Code Cells

Code cells have execution counts and output

- Are specific to the language (kernel) being used
- Run the code *in the order of cells run* not location

3.6 Embeds

You can embed HTML using code cells based on the language.

- Python:

```
from IPython.display import YouTubeVideo
YouTubeVideo("E7LlvXQPbvY", width=400)
```

- R:

```
IRdisplay::display_html("<embed code from Youtube>")
```

3.7 Self-Tests

- Include a supplemental file with a `tests()` function
- Call the function to evaluate for an answer.
- Hash the answer and correct response to avoid peeking at solutions

3.8 R Example

In Notebook:

"What year was the Magna Carta first signed?"

```
# fill in the answer
source("tests.r")

year <- ???

test1(year)
```

In `tests.r`

```
test1 = function(year) {

if(year == 1215){
  print("Success")
} else {
  print("Try Again")
}
```

```
}
```

3.9 Hashed Version

```
library(digest)

test1 = function(year) {

  if(digest(year) == "1f507200cb6c053bf84794bee409b202"){
    print("Success")
  } else {
    print("Try Again")
  }

}
```

3.10 Python Example

In Notebook:

"What year was the Magna Carta first signed?"

```
# fill in the answer
import tests.py as t

year = ???

t.test1(year)
```

In `tests.py`

```
def test1(year):
  if year == 1215:
    print("Success")
  else:
    print("Try Again")
```

3.11 Hashed Version

```
import hashlib

def hash_it(obj):
    return int(hashlib.sha1(obj.encode('utf-8')).hexdigest(), 16)

def test1(year):
  if hash_it(str(year)) == 360150900849359670614763641985028241730513070521:
    print("Success")
  else:
    print("Try Again")
```

4 Addendum: Sharing and Deploying

4.1 Sharing Notebooks

The "final mile" in curriculum develop is sharing your work with students so they can use it. There are basically two main options:

1. Upload your completed notebooks and any other files to Canvas (or another website or LMS) then have students download them, then upload them to the JupyterHub you are using.
2. Upload your completed notebooks and any other files to GitHub (or another repository host) then share using `nbgitpuller`

There are benefits and costs to both options

4.2 Option 1

- The key benefit to Option 1 is that it can be completely private (it's all on Canvas)
- It's also simple, since no new tools or workflows are involved

The main cost is that if you have many files it can be time consuming and hard to manage, especially if you need to change and update things
* File management on a Jupyterhub is also not completely intuitive unless you are fairly familiar with Linux, and your students probably won't be.

4.3 Option 2

Most people opt for Option 2, which uses a public GitHub Account and the software `nbgitpuller` which is installed on the JupyterHub. Conceptually, it is fairly simple:

- Create a public GitHub repository and upload your notebooks and files to it
- Use `nbgitpuller` to generate a link to your repository
- Click the link to load the notebooks

It does add an extra step (GitHub) and requires you to post you material publicly, which may be undesirable for some uses.

4.4 `nbgitpuller`

This is a small application which loads repositories into JupyterHubs. You can try it out on the COMET Website by clicking on "Launch."

- The documentation is a good first step but realistically all you really need to do is generate a link
- They supply a validated generator here: https://nbgitpuller.readthedocs.io/en/latest/link.html

---

If you're using Jupyter Open:

- Put `https://open.jupyter.ubc.ca/jupyter/hub` in the *JupyterHub URL* field
- Copy your GitHub repository's URL into the *Git Repository URL* and select the appropriate branch (`main` if its a new repository)
- You don't need to select a file to open, but if you want the link to open a specific file, indicate the path within the repository
- You can also select the environment; generally you should choose *JupyterLab* or *Classic Jupyter Notebook* which is a little simpler-looking

You can test the generated link by copying it into a browser to see if it works.

4.5 Sharing and Updating

You can now share the link with students (a link shortner helps) to load your files

- This create a *copy* on their own account on the hub
- If you want them to refresh it, they have to delete the current copy on their own account

- This can be tedious; it's easiest to use the terminal on the hub and do `rm -r <directory name>`

That's it! Happy computing!

5 Resources

5.1 Tutorials and Training

- Markdown Reference Guide
  - A Super Simple Tutorial on Markdown
- A Tutorial For How to Install Locally (Hard!)
- A JupyterLab Tutorial
- Quarto Getting Started
- Berkley Jupyter Resource Library

5.2 Docs

- IRKernel Reference: want to use R locally? Here you go!
- STATA Kernel Reference: want to use STATA locally? Here you go!
- Jupyter Docs
  - JupyterLab Docs

5.3 Examples at UBC

- QuantEcon
  - In particular, this course
- COMET
- DSCI100
- STAT201

5.4 References

Barba, Lorena A, Lecia J Barker, Douglas S Blank, Jed Brown, Allen B Downey, Timothy George, Lindsey J Heagy, et al. 2019. "Teaching and Learning with Jupyter." *Recuperado: Https://Jupyter4edu. Github. Io/Jupyter-Edu-Book*.

Granger, Brian E, and Fernando Pérez. 2021. "Jupyter: Thinking and Storytelling with Code and Data." *Computing in Science & Engineering* 23 (2): 7–14.

Johnson, Jeremiah W. 2020. "Benefits and Pitfalls of Jupyter Notebooks in the Classroom." In *Proceedings of the 21st Annual Conference on Information Technology Education*, 32–37.

Perkel, Jeffrey M. 2018. "Why Jupyter Is Data Scientists' Computational Notebook of Choice." *Nature* 563 (7732): 145–47.

Randles, Bernadette M, Irene V Pasquetto, Milena S Golshan, and Christine L Borgman. 2017. "Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study." In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 1–2. IEEE.