OpenScience

October 15, 2019

1 Open Science: Theory and Practice

1.1 Overview and Objectives

"Open Science" is something of an umbrella term encompassing everything related to reproducible, transparent science.

There have been some complaints that Open Science is amorphous and ambiguous, that its prescription for reproducibility is not, in itself, reproducible.

However, Open Science is broadly defined and meant to appeal to every area of science, from life sciences to computational sciences to theoretical sciences. What Open Science looks like is field-specific, but there are general principles that cut across all fields of science.

By the end of this lecture, you should be able to

- Define "open science" and its importance to scientific inquiry
- Recall the core strategies for reproducing and replicating results
- Connect open science practices to case studies in research misconduct

1.1.1 Case Study 1 (Andie Anderson)

Dr. Bob's major clinical research project is a prospective, longitudinal study of changes over time in plasma levels of protein X and their association with cardiovascular disease. If it is successful it would be publishable in a high-impact journal and give a substantial boost to his achieving tenure.

Dr. Miriam is offered an opportunity to analyze data from the first 3 time points of his protein X study. She performs a statistical analysis on a spreadsheet provided by Dr. Bob, but her results are not consistent with the hypothesis Dr. Bob wrote in his grant applications as she found no association of Protein X with cardiovascular risk. When Dr. Miriam presents her analysis to Dr. Bob, he is noncommittal and suggests that she has incorrectly analyzed the data. He says he will check her work and the next week, Dr. Bob returns the spreadsheet to Dr. Miriam, explaining that he has corrected a few mistaken data entries. He asks her to redo the analysis.

Office of Intramural Research. (2014). Differentiating between Honest Discourse and Research Misconduct. Retrieved from https://oir.nih.gov/sites/default/files/uploads/sourcebook/documents/ethical_conduct/case_studies-2014.pdf

1.2 Part 1: What is "Open Science"?

Simply put, Open Science is the **movement to make all scientific data, methods, and materials accessible to all levels of society**.

Why is this a good thing?

- The vast majority of research is publicly funded; it would seem logical that the public have access to it!
- Open and transparent research makes peer review and replication much easier.
- There is some convincing evidence that Open Science gives projects more downstream impact in the scientific community

Nonetheless, there are some downsides to making everything openly available.

- The deluge of science will overwhelm already heavily-burdened researchers.
- The tools could be used for more nefarious purposes (influenza strain, GPT-2 language model).

Work resumes on lethal flu strains

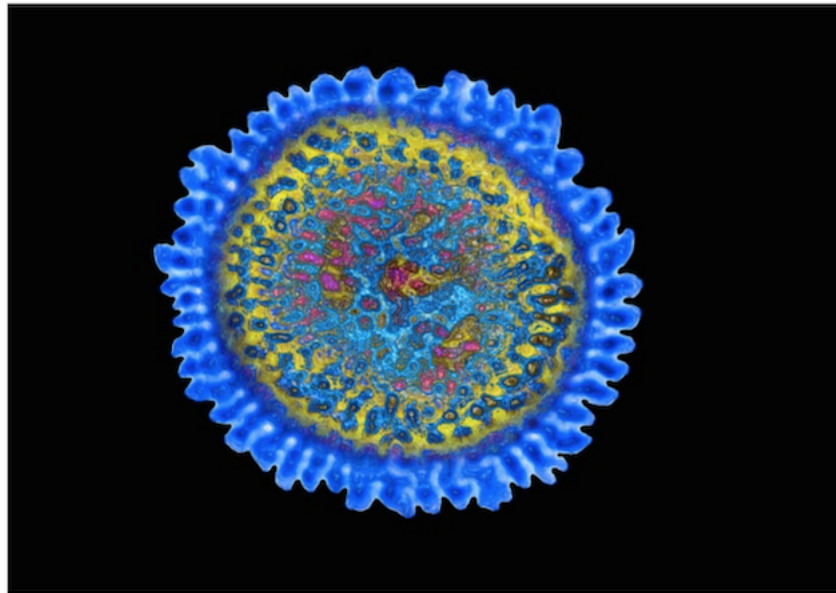**Study of lab-made viruses a 'public-health responsibility'.**

Declan Butler

23 January 2013

PDF   Rights & Permissions

JAMES CAVALLINI/SPL

A moratorium on research that modifies the potential virulence of avian influenza virus has now been lifted.

An international group of scientists this week ended a year-long moratorium on controversial work to engineer potentially deadly strains of the H5N1 avian flu virus in the lab.

My opinion–as you've probably guessed by the title of the lecture–is that the benefits of good Open Science practices outweigh the drawbacks, for the following reason:

I learn best when I can dig in and get my hands dirty.

Reading a paper or even a blog post that vaguely describes a method is one thing. Actually seeing the code, changing it, and re-running it to observe the results is something else entirely and, so I believe, is vastly superior in educational terms.

The scientific deluge is legitimate, though this was already happening even without the addition of open data, open access, and open source. And it would seem that, while we do absolutely need to exercise caution in our research and not pursue the ends by any means necessary, science is the pursuit of knowledge for its own sake and that should also be respected to the highest degree.

To that end, there are **six** main themes that comprise the Open Science guidelines.

1. Open data: all data used in the project should be made available.
2. Open source: all code written in the project should be publicly available.
3. Open methods: the exact procedure of the project is publicly documented.
4. Open review: correspondence between reviewers and authors is public.
5. Open access: resulting publications are publicly available.
6. Open education: all education materials are publicly available.

1.2.1   1: Open Data

If you had to pick the "core" of Open Science, this would probably be it. All of the data used in your study and experiments are published online.

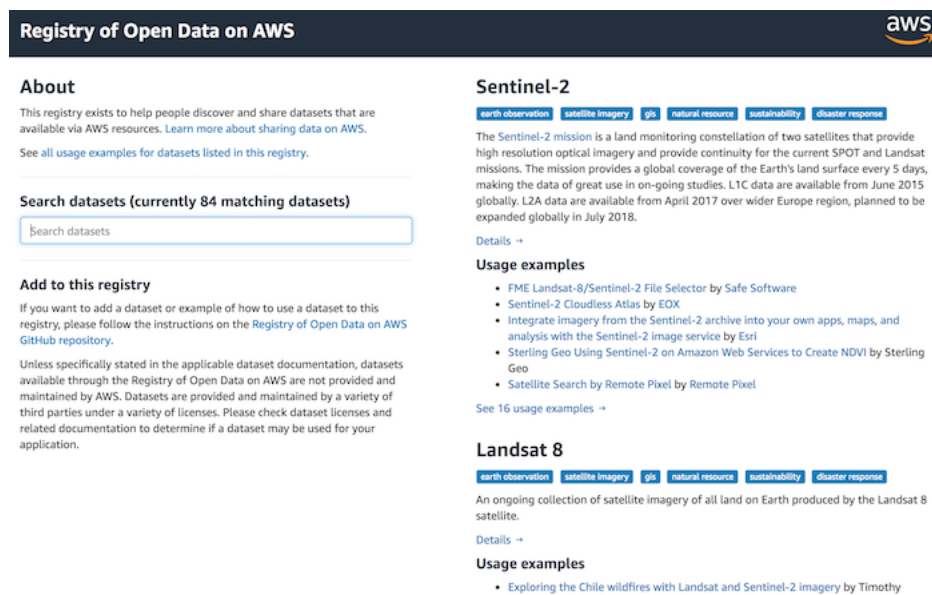This is definitely a shift from prior precedent; most raw data from scientific experiments remain cloistered.

The situation is further complicated by Terms of Service agreements that prohibit the sharing of data collected. - For example: if you hooked up a Python client to listen to and capture public Twitter posts, you are forbidden from sharing the Twitter data publicly. - Which seems odd, given that the data are public anyway, but there you go.

Repositories and online data banks have sprung up around this idea. Many research institutions host their own open data repositories, as do some large tech companies.

- CERN, the organization behind the Large Hadron Collider, has posted its data online: http://opendata.cern.ch/about/CMS
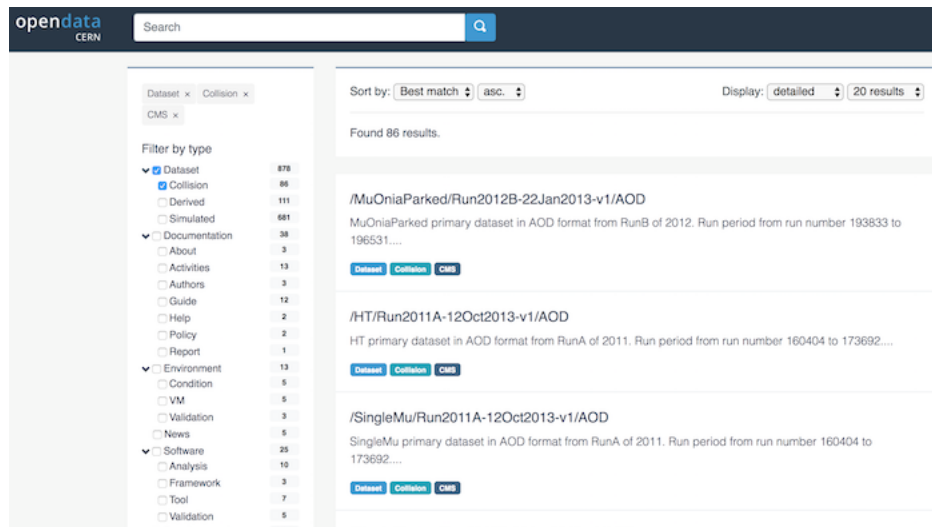
- Amazon has released its own set of large public datasets: https://aws.amazon.com/public-data-sets/

- Kaggle also has some pretty fantastic open datasets from its competitions: https://www.kaggle.com/datasets

• DataHub is a general-purpose repository for anyone to submit their own datasets. https://datahub.io/en/dataset
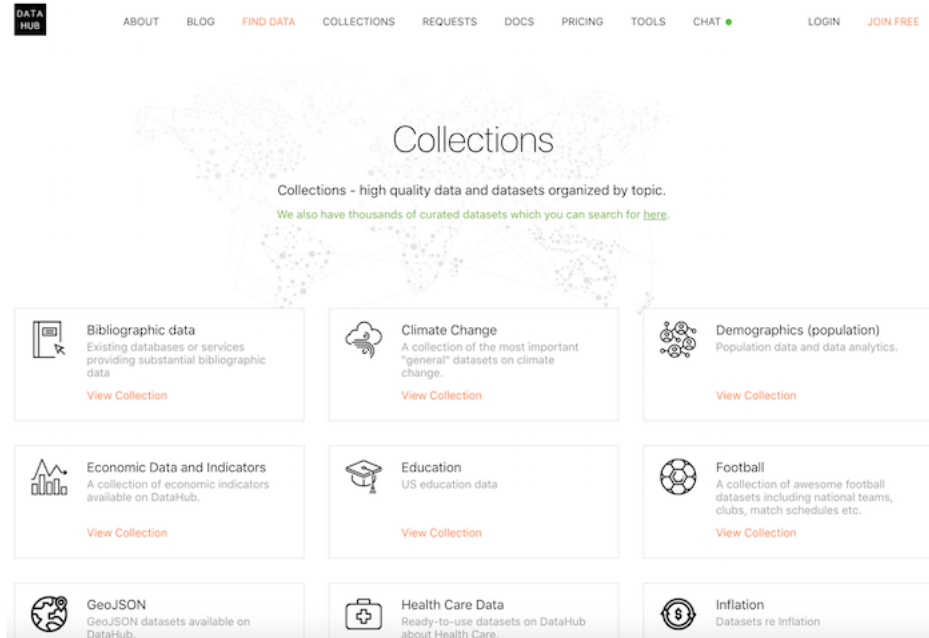
I encourage you to google "open datasets" in your area of interest! Go check them out!

1.2.2   2: Open Source

This is probably the part you're most familiar with. Any (and *all*) code that's used in your project is published somewhere publicly for download.

There are certainly conditions where code can't be fully open sourced–proprietary corporate secrets, pending patents, etc–but to fully adhere to Open Science, the code has to be made completely available for anyone.

Like with open data, there are numerous **repositories** across the web that specialize in providing publicly-available versioning systems for both maintaining and publishing your code.

- GitHub is easily the most popular, and is where the materials for this lecture are published! https://github.com/

- BitBucket is another option that also uses git to manage team codebases https://bitbucket.org/

- SourceForge is one of the oldest and most well-known online repositories https://sourceforge.net/

If you're looking for the actual source code, it's pretty likely to be on one of these sites.

1.2.3   3: Open Methods

This is probably the trickiest item. How does one make *methods* reproducible?

Open source code is part of it, but even more important is the effort put into making the methods in the code understandable. This takes several forms:

- Documentation, both as accompanying doc files (e.g. JavaDoc) as well as in-code comments
- Proofs of the methods devised if they're novel, or references to their original sources
- Pre-packaged examples that will run with little or no prior configuration on the user's part
- Pre-registered methods *before* any work is conducted
- Self-contained virtual container scripts that satisfy all the prerequisites

1.2.4   Case Study 2 (Ummay Mowshome Jahan)

Hwang Woo-Suk, a South-Korean researcher, started his career as a scientist after getting his PhD at Seoul National University (SNU), and specialized in the reproduction of cattle. Working with in vitro fertilization and cloning of animals for years at SNU, he became a famous scientist in South-Korea, claiming to have cloned cows and pigs and genetically engineering a pig to produce human organs (among other accomplishments). He then began research on stem cells, and acquired powerful friends in the government, most notably the president himself, apparently demonstrating how he cured a sick dog using stem cell treatment. He also gained popularity in the public by appearing in media with sensational demonstrations. This led to his research being heavily financed, through government funds and donations from the public. In 2004 and 2005, articles were published by Hwang and colleagues, and these were the first times anyone could seemingly prove having created stem cells from cloning. These articles, being published in the highly respected journal Science, and the fact that he had collaborated with Dr. Gerald Schatten (a famous American scientist) on the two papers, was evidence for his earlier unpublished experiments.

Then, after a year, Hwang published an article in Nature describing the cloning of an Afghan dog, named Snuppy. Hwang was already a national hero in South Korea. South Korea was now regarded as the leading country of stem cell research, and the Korean people placed high expectations for him and his work. A survey showed that 30% of women said they would donate egg cells for Hwangs research.

However, already in 2004, an article in Nature reported that two junior scientists working with Hwang had donated eggs for the for the 2004 Science paper. Then, a South Korean investigative TV-show started doing research about his methods, reported that he had forced these junior scientists into donating their eggs. And that DNA-test showed that of one of the cell lines in the 2005 paper did not match the DNA of the somatic tissue donor. Science reported that a review board for Seoul National University in an investigation found out that donors had received payment for the eggs. Then, things started to fall apart. Following accusations of fabrication, Hwang informed Science that some of the pictures in the 2005 article were taken from another article (the pictures were later admitted being faked). One of Hwangs collaborators then told a news network that

"there are no cloned embryonic stem cells". Hwang and Dr. Schatten later retracted their article from Science, stating that analyzing the data had led to the conclusion that "the results could not be trusted". An investigation of Hwangs work by Seoul National University concluded that large amounts of the 2005 Science paper had been fabricated.

Hwang undoubtedly made important contributions to science and stem cell research. The cloning of Snuppy appears to be genuine, and his nuclear extrusion technique was also used by Dr. Schatten and colleagues successfully cloning a monkey. The Hwang affair is therefore a remarkable story about a talented researcher that ended up wasting his career and talent.

1. What types of research misconducts are present in this case?
2. What type of consequence researchers can face after this type of research misconduct is occurred?

- https://stemcellswebzine.wordpress.com/2011/04/26/the-hwang-scandal/
- https://www.nature.com/news/2005/051219/full/news051219-
  3.html?utm_source=commission_junction&utm_medium=affiliate

1.2.5   4: Open Review

The cornerstone of the scientific process is that of *peer review*: your peers, your colleagues, your fellow researchers should vet your work before it's officially included as part of the scientific literature.

However, this process is fraught with ambiguity and opacity. Conflicts of interest can potentially lead to biased reviews (if you're reviewing the paper of a competitor, it isn't exactly in your best interest to go easy on them), and it can be difficult to assess a published paper in the public sphere without a trail of edits from which to begin.

Online open review journals such as **The Journal of Open Source Software (JOSS)** have begun proliferating to address this shortcoming.

Reviews essentially take the form of GitHub tickets, and researchers in the field can discuss and debate the merits of the work in an open forum.

There's also a site, called "Open Review", where conferences can elect to have their papers openly and publicly reviewed, threaded-comment style.

Would you be amenable to having your publication reviewed publicly, where both your reviewers' feedback and your responses are publicly viewable?

1.2.6   5: Open Access

Once a project is published, the paper should be made publicly available for anyone, anywhere to download and read for themselves.

Easily the most popular open access paper repository is **arXiv** (pronounced "archive"…geddit?). Other repositories modeled directly after its success, such as bioRxiv, have already started springing up.

arXiv is already *hugely* popular.

Open access to 1,169,533 e-prints in Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance and Statistics

In fact, so many papers are archived here on a regular basis, that someone created their own open source "aggregator" service: http://www.arxiv-sanity.com , which collates the papers you want to read and helps filter out all the others.

Preprints have become *so* popular, that versions of arXiv have popped up for almost every area!

If you're already posting your papers on your own personal website, give arXiv a try! Some of them even have automated journal submission buttons baked into the site!

1.2.7   6: Open Education

This and Open Methods are closely related. In this sense, any course materials that come from research that is done are made available for others.

MIT OpenCourseWare is probably the best example of open education at work, but these types of sites are proliferating.

- Many courses are making their materials freely available on GitHub (this lecture is also freely available!).

- Tech companies such as GitHub (aptly named: GitHub Classroom) are developing educational tools specifically aimed at student classrooms. Amazon Web Services provides a similar platform for course materials, often bundled with its cloud compute services.

Note: This does not by default include MOOCs. If MOOCs make their materials freely available online, then and only then would it fall under this section.

So that's all great! But...

What can *I* do? Where do I start?

1.2.8 Case Study 3 (CT Peterson)

Professor Powers ran a laboratory where post-doctoral student Harrison Green works. Recently, he had been attempting to develop a new anti-sense technique that would more easily enable one to shut down the expression of a particular gene. At first, Green was having difficulty getting his technique to work. Suddenly, during a laboratory meeting, Green announced that he had gotten his technique to work and produced data showing statistically significant results.

About a week later when Powers was looking for some scrap paper, she found a random piece of paper with data recorded on it. The data appeared to be almost identical to those presented by Green and indeed, she recognized the writing as belonging to Green. She immediately noticed that the scratch paper had an extra data point that Green had apparently deleted from the treatment group. With that extra data point included, Green's results appeared no longer to be statistically significant. Indeed, this very fact was noted on the scratch paper: "$p > .05$. No Good."

The next day, Powers asked Green why he had deleted the data point. Green responded that it was an outlier and had been deleted because he had botched the experiment that day and felt justified in tossing the point out. Powers asked to see the raw data for that experiment, but when Green gave her his lab notebook, pages describing the experiment were clearly missing.

That evening, while pondering what to do, Powers reanalyzed Green's data and found that Green had miscalculated the t value, and that even with the discarded treatment value, Green's original results were actually statistically significant.

Do you think falsification of results occurred? Why or why not? What should Powers do, if anything?

References: https://www.medschool.umaryland.edu/media/SOM/Offices-of-the-Dean/Office-of-Research/documents/charrow_lecture_handout.pdf

1.3 Part 2: Open Science Best Practices

In the previous section we went over the absolute essentials for an Open Science project or research endeavor. It included some examples of real-world services and tools to help expedite the process.

Here, we'll go into a bit more detail as to exactly how you should tweak your projects to be truly Open Science. It's not really something you can just "tack on" at the end; rather, you'll need to design your projects from the onset to be part of the Open Science initiative.

1.3.1 Version control

This is a good starting point for any project.

Version control keeps track of changes within your working documents.

It's a lot like Google docs, but for any kind of file–documents, spreadsheets, text files, program code.

You all know Dropbox? Has it ever saved your life? If so, the next time you start a coding project, use some form of *version control*.
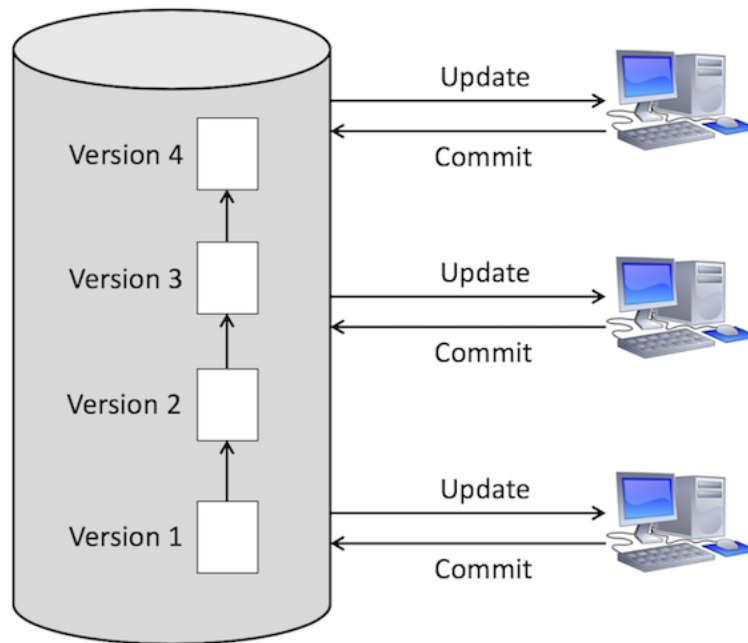
There are several different version control systems out there for code, depending on what you like.

I personally prefer git, but I've used all of these before.

A lot of these have graphical interfaces you can use if you really want to avoid the command line!

1.3.2 Anatomy of an open source project

There are some common files and folder hierarchies to implement at the very start of a new project. The shablona GitHub repository has all this documented very well, but here are the highlights:

- A `README` file of some kind. This is your introduction that summarizes your project, gives a basic overview of how to use it, what the prerequisites are, links to further (more detailed) documentation, and any problems that are known.

- Your `src` folder, containing all your source code.

- A directory of `tests` for your code. Always always always *always* **always** write tests!

- A `doc` folder containing *detailed* documentation about every aspect of your project, including any examples.

- A build script of some sort: a script you can run which automates the majority of the install process.

- Continuous integration (CI). This is bit more advanced, but the idea of continuous integration is to have your code exported out to an auto-compiler every time you update the code, so that you know *immediately* whether or not it builds successfully.

- A `LICENSE` file. This is often overlooked (like clicking through a EULA), but in the age of open source and Open Science it is *absolutely essential* to have a license of some sort.

```
shablona/
    |- README.md
    |- shablona/
        |- __init__.py
        |- shablona.py
        |- due.py
        |- data/
            |- ...
        |- tests/
            |- ...
    |- doc/
        |- Makefile
        |- conf.py
        |- sphinxext/
            |- ...
        |- _static/
            |- ...
    |- setup.py
    |- .travis.yml
    |- .mailmap
    |- appveyor.yml
    |- LICENSE
    |- Makefile
    |- ipynb/
        |- ...
```

1.3.3 Document, document, document

This is drilled into computer science students from day 1, and yet one constant across every software project is that the *documentation is terrible*. To meet basic adequacy standards, documentation should include

- **The code itself**. This is the radical notion that code should be easy to understand, especially if read by someone who did NOT write it.

- **Comments in the code**. Invariably there will be some logic in the code that doesn't make sense at first glance. This should be explained with clear comments, preferably with some examples.

- **README**. This is usually the user's first introduction to a new software package (GitHub makes this file the default landing page of a repository!). It should clearly state the purpose of the software, how to install and run it, and some basic usage examples before pointing the user to where they can go to learn more.

1.3.4 Examples and tests

Right after the README, this is the next place the user will go: examples for how to run the code. Is it command line? What options are there? What kind of data can I run through it? What should I expect for output? These sorts of questions can be addressed with

- **Quickstart**. This is file that is dedicated to getting the user up and running with the software as quickly as possible with an example. The goal is to put the user in the driver's seat as quickly as possible, and let them ask questions later.

- **Examples**. This usually takes the form of a bash script, written to invoke the code and have it run on a sample dataset that's provided. Alternatively, you can give a step-by-step guide of what commands to run, what inputs to give, and what the expected output should be.

- **Unit tests**. You'll learn more about this as you take higher-level software engineering courses, but the main idea is you want to write code that *tests* your main code. The user can also go here to get a feel for how the code should be used.

- **Containers**. This is a very advanced topic that we won't cover in this course, but the "wave of the future" is to build your software inside *containers* (e.g. Docker). Think of containers as super-lightweight virtual machines that anyone can build from a simple script. Using containers, you can package your software in such a way that you don't have to worry about operating systems or clashing software dependencies.

Containers, such as **Docker**, allow you to create *recipes* that build the precise environment you want, from the operating system all the way up to the exact version of OpenOffice you want.

This has *huge* implications in reproducibility: you can provide others with an exact set of instructions–even a pre-built "snapshot"–of the environment in which you conducted your work.

### 1.3.5 Scientific Notebooks

If you're an R or Python user, you've probably heard of "scientific notebooks".

They're interactive, usually web-based applications that blur the lines between - written documents - code modules - figures and tables

by interleaving all three within the same medium.

**These very slides were built entirely in Jupyter, a multi-language scientific notebook platform!
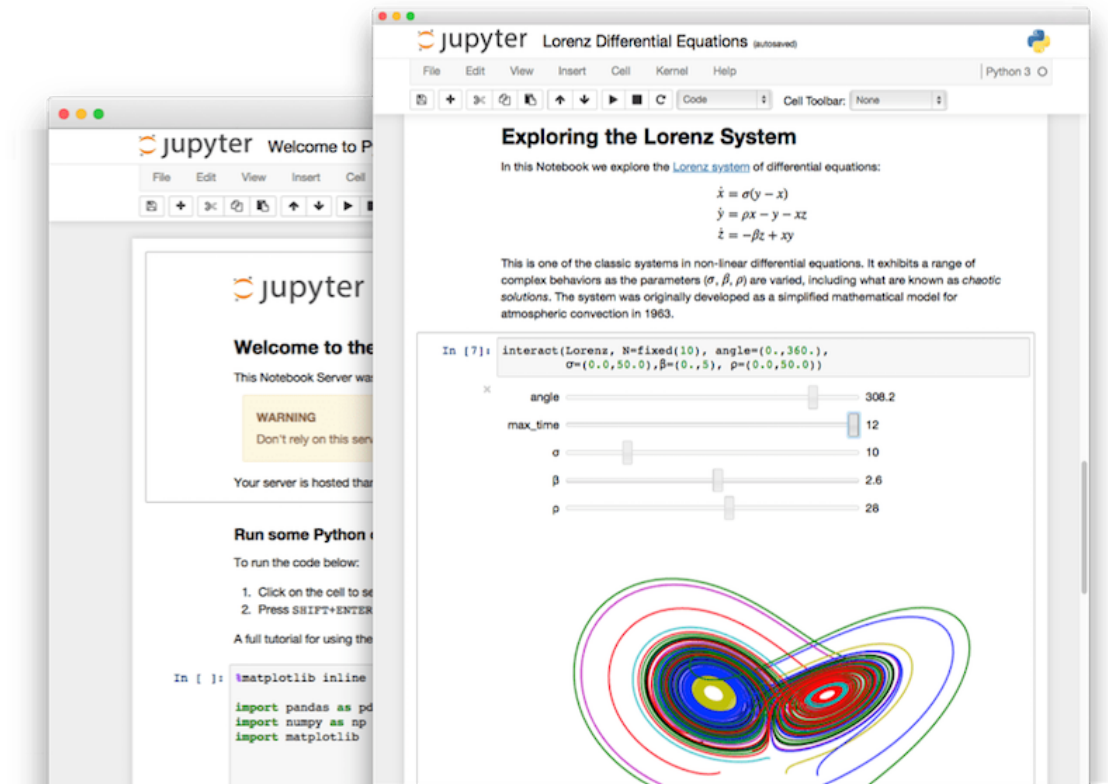
While these slides were built in Jupyter, they were exported into HTML and PDF, so the code isn't technically live anymore.

However, through platforms like mybinder, you can turn any GitHub repository into a collection of active notebooks! You can do this yourself with these slides–all you need is a web browser!

1.3.6   Study pre-registration

Pre-registration means *publishing your methods publicly BEFORE any work has been done*.

Why is this important?

Sites like Protocols.io https://www.protocols.io/ and OSF https://osf.io/prereg/ have pre-registration modules, sometimes linked with certain journals who agree to publish the results

of the methods, regardless of whether they're positive or negative.

1.3.7 Licensing

Yes, this is super boring, I agree. Fortunately, the wonderful folks at GitHub have created an awesome resource for you to use.

Choose a License: It provides a handy choose-your-adventure flowchart for picking the perfect open software license for your project. I highly recommend it.

However, if you're just looking for a **tl;dr** version and want the basic lay of the land, GitHub's default licenses for new projects include

- MIT License (pretty much the "no warranty whatsoever, use at your own risk" license)
- Apache 2.0 License (almost exactly like the MIT, but allows for patents and trademarks)
- GPLv3 License (requires anyone who modifies your code to publish those modifications under GPLv3 too)

1.4 Final Thoughts

In my view, Open Science and its practices will only become more important. - Replicating previous results - Providing hands-on tools to the upcoming generation of scientists - Making cutting-edge resources accessible to broad audiences

Six major areas of Open Science - These have to be addressed from the beginning, not tacked on after the fact

Already lots of great open source tools and repositories to encourage Open Science practices - data repositories - tools for method replication and testing - standards for code structure, results presentation, and method preregistration

2 Questions?

2.1 Additional Resources

These slides! https://github.com/quinngroup/openscience-intro-oct2019

1. GitHub template for open source Python projects https://github.com/uwescience/shablona
2. Mozilla Open Science Lab https://science.mozilla.org/
3. Open Science Framework (OSF) for all phases of the scientific pipeline https://osf.io/
4. The Center for Open Science (COS), which supports OSF https://cos.io/
5. Docker https://www.docker.com/