Day 2

Outline

1. Overview of the too
2. Intro to Jupyter
3. Navigating yo
4. oject set
5. e t elp
6. on
7. Goo e etiquette

Day 1

8. Data processing with pandas
9. Automation with python
10. Reproducibility
11. Version control

Day 2

Intro to Pandas

Open the notebook called:
06_Intro_to_pandas.ipynb

Intro to Automation

Open the notebook called:
07_Intro_to_automation.ipynb

Day 2 – Reproducibility

Reproducibility

What is it?

The ability for you, or others, to redo your work in order to:

1. Check for correctness (confirmation)
2. Use the same method on new data (adaptation)
3. Use a different method on the same data (modification)

Is reproducibility important?

The reproducibility crisis in psychology

RESEARCH ARTICLE

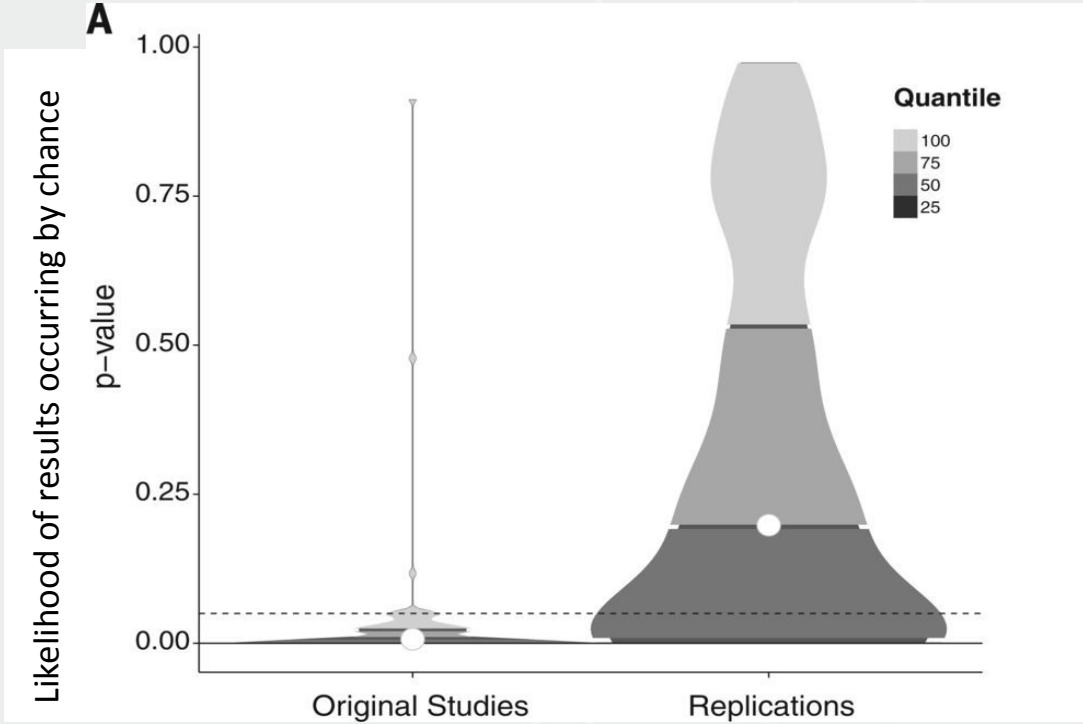
Estimating the reproducibility of psychological science

Open Science Collaboration*,†

+ Author Affiliations

†Corresponding author. E-mail: nosek@virginia.edu

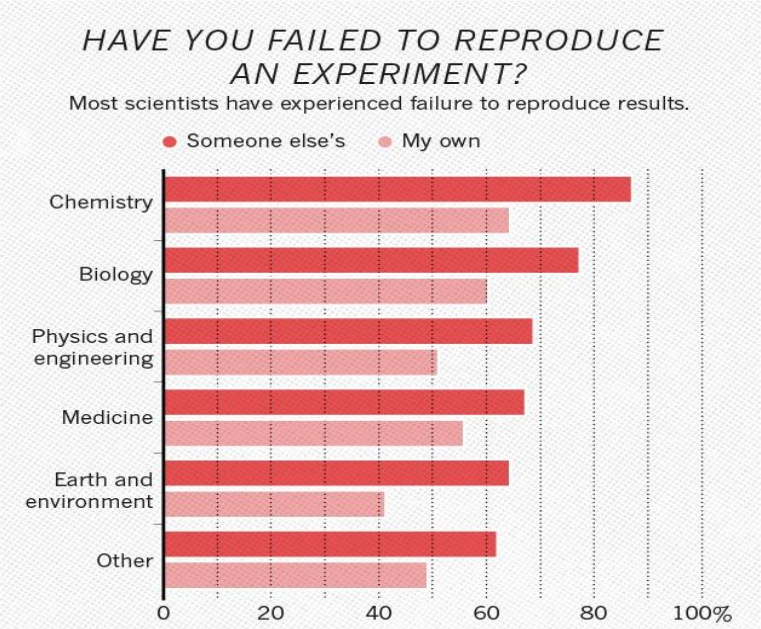
Science 28 Aug 2015:
Vol. 349, Issue 6251,
DOI: 10.1126/science.aac4716

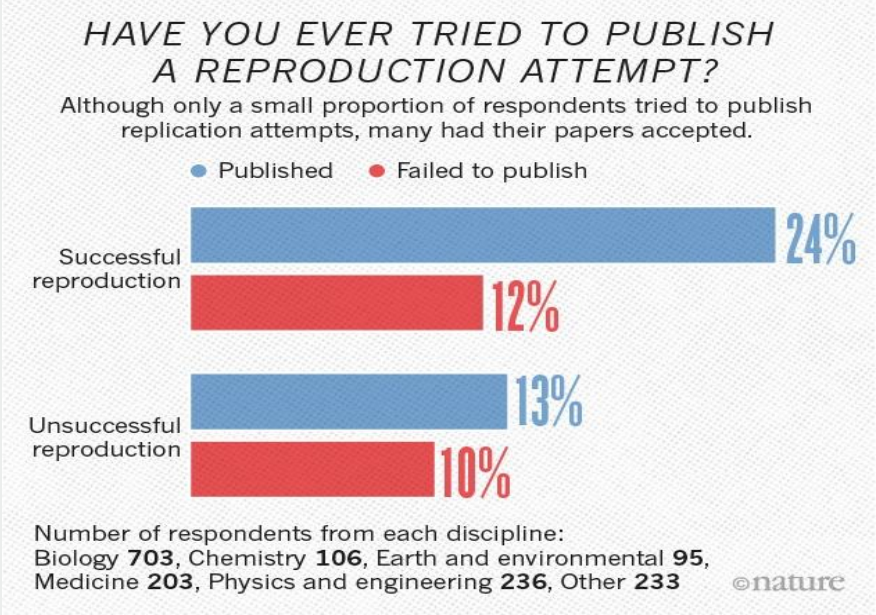


http://dx.doi.org/10.1126/science.aac4716

We should all be worried about reproducibility

Science is based on reproducible results, but many results cannot be reproduced.





https://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970

Reproducibility

Why is it important?

If you cannot recreate your results with the same data and same method then you should not trust your results.

Confirmation is important for building confidence and trust.

Adaptation allows you to reuse methods that you are confident with.

Modification is a natural part of the development process and ultimately saves time.

Scenario - You predict a big loss/gain for your area

5:00 Stop

You want to be sure of your results before you present them

Rerunning your analysis from a fresh start and getting the same result builds confidence

How do you achieve a “fresh run” of your analysis?

1. Full automation
2. Human readable recipe
3. Scripted process
4. Start again and hope that you can find all the relevant post-it-notes

Be brave and share your experience

Scenario - You predict a big loss/gain for your area

You want to be sure of your results before you present them

Rerunning your analysis from a fresh start and getting the same result builds confidence

How do you achieve a “fresh run” of your analysis?

1. Full automation
2. Human readable recipe
3. Scripted process
4. Start again and hope that you can find all the relevant post-it-notes

A year later your predictions don’t pan out.
Can you still defend your analysis, and identify where your model diverged from reality?

Scenario - Loss of personnel

A key person in your org has moved to another group/company.

This person was responsible for generating reports that are critical to your business. They use a semi-automated system, and it normally takes them 30 mins to complete a report.

- Could you continue their work?
  - Where is the data, code, info on what they are doing?
- How long would it take you to run their reporting?
- How can you improve this scenario?

Does your process pass the lotto factor?

Documentation

Documentation is key to reproducibility

- Documentation is the idea of documenting your procedures for your experiment/process so that an outsider could understand the workings of your team
  - The audience is a user
- Remember the Lotto factor
- Documentation is a love letter to your future self - Damian Conway

Nomenclature and Naming Conventions

Language is important, precise language even more so

- Clear and concise language aids transparency
- Standard names and terminology makes language accessible
- Avoid jargon/TLAs where possible
  - Describe/introduce when necessary
  - A reference document for terminology is useful
- Apply to all communication channels

Nomenclature and Naming Conventions

Language is important, precise language even more so

- File [+ Program/Process/Function/Variable] naming should be consistent
  - Team members won't have to over think the naming process
- Easier to facilitate access, retrieval and storage of files
- Easier to browse through files saving time and effort
- Harder to lose!

Case study

Former PhD student and subsequent founder of the Figshare platform, Mark Hahnel, typified a common challenge:

"*During my PhD I was never good at managing my research data. I had so many different file names for my data that I always struggled to find the correct file quickly and easily when it was requested. My former PI was so horrified upon seeing the state of my data organisation that she held an emergency lab book meeting with the rest of my group when I was leaving.*"

Research Information, April/May 2014

Example

As previously suggested, consistent and meaningful naming of files and folders can make everyone's life easier. See this example below:

YYYYMMDD_SiteA_SensorB.CSV to encode Date Location Sensor

Which when applied, would look like this below:

20150621_Yaouk_Humidity.CSV

Some characters may have special meaning to the operating system so avoid using these characters when you are naming files. These characters include the following: / \ " ' * ; - ? [ ] ( ) ~ ! $ { } < > # @ & | **space** tab newline

Discussion

Good or Bad names

1. Myfile.docx
2. 20200130_core_instructions_draft.docx
3. Fig 2.png
4. 20200130_core_instructions_v1.docx
5. My figure (3) v2.png
6. fig02_core_instructions_presentation.png

https://speakerdeck.com/jennybc/how-to-name-files?slide=3

Ten Simple Rules for Reproducible *Computational* Research

1. For every result, keep track of how it was produced
2. Avoid manual data manipulation steps
3. Archive the exact versions of all external programs used
4. Version control all custom scripts
5. Record all intermediate results, when possible in standardized formats
6. For analyses that include randomness, note underlying random seeds
7. Always store raw data behind plots
8. Generate hierarchical analysis output, allowing layers of increasing detail to be inspected
9. Connect textual statements to underlying results
10. Provide public access to scripts, runs, and results

From http://dx.doi.org/10.1371/journal.pcbi.1003285

Ten Simple Rules for Reproducible *Computational* Research

1. For every result, keep track of how it was produced
2. Avoid manual data manipulation steps

“

As a **minimum**, you should at least record sufficient details on programs, parameters, and manual procedures to allow yourself, in a year or so, to approximately reproduce the results.

…

If working at the UNIX command line, manual modification of files can usually be replaced by the use of standard UNIX commands or small custom scripts.

”

Do by hand to learn/test/explore, then encode this in a script or instruction set that future you, and future others, could understand and run

From http://dx.doi.org/10.1371/journal.pcbi.1003285

Ten Simple Rules for Reproducible *Computational* Research

1. For every result, keep track of how it was produced
2. Avoid manual data manipulation steps
3. Archive the exact versions of all external programs used
4. Version control all custom scripts
5. Record all intermediate results, when possible in standardized formats
6. For analyses that include randomness, note underlying random seeds
7. Always store raw data behind plots
8. Generate hierarchical analysis output, allowing layers of increasing detail to be inspected
9. Connect textual statements to underlying results
10. Provide public access to scripts, runs, and results

From http://dx.doi.org/10.1371/journal.pcbi.1003285

Ten Simple Rules for Reproducible *Computational* Research

1. For every result, keep track of how it was produced
2. Avoid manual data manipulation steps
3. Archive the exact versions of all external programs used
4. Version control all custom scripts
5. Record all intermediate results, when possible in standardized formats
6. For analyses that include randomness, note underlying random seeds
7. Always store raw data behind plots
8. Generate hierarchical analysis output, allowing layers of increasing detail to be inspected
9. Connect textual statements to underlying results
10. Provide public access to scripts, runs, and results

From http://dx.doi.org/10.1371/journal.pcbi.1003285

Ten Simple Rules for Reproducible *Computational* Research

1. For every result, keep track of how it was produced
2. Avoid manual data manipulation steps
3. Archive the exact versions of all external programs used
4. Version control all custom scripts
5. Record all intermediate results, when possible in standardized formats
6. For analyses that include randomness, note underlying random seeds
7. Always store raw data behind plots
8. Generate hierarchical analysis output, allowing layers of increasing detail to be inspected
9. Connect textual statements to underlying results
10. Provide public access to scripts, runs, and results

From http://dx.doi.org/10.1371/journal.pcbi.1003285

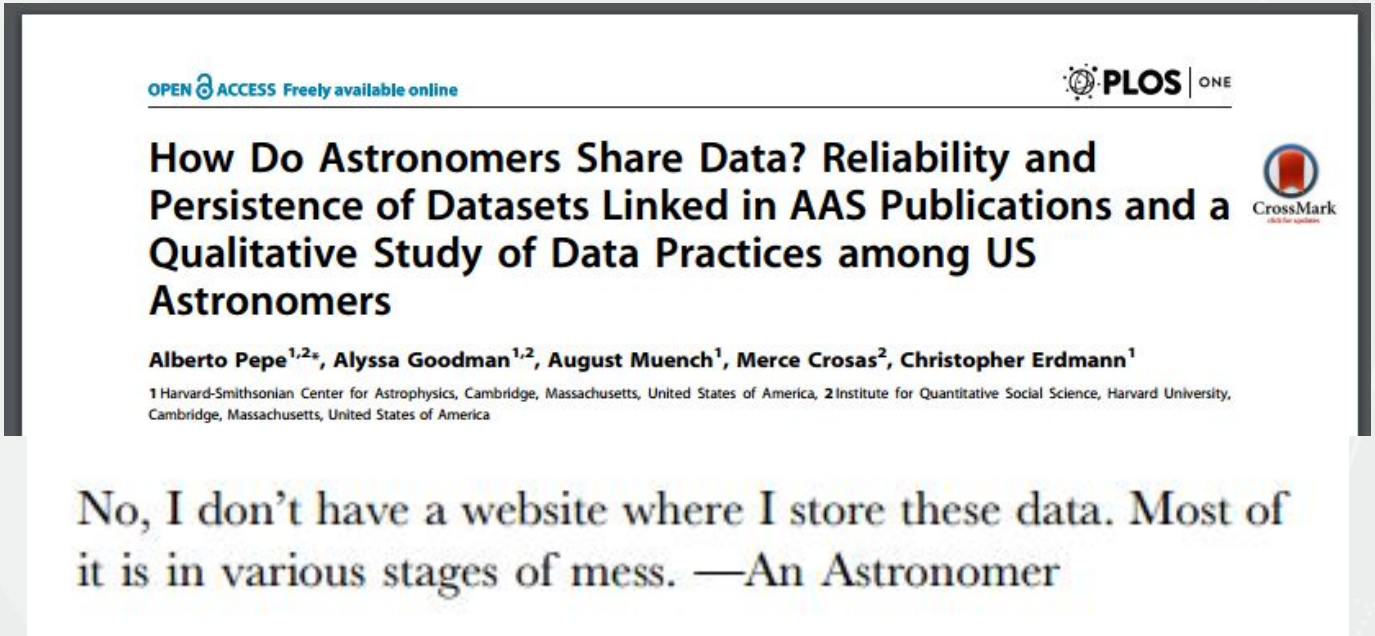
Ten Simple Rules for Reproducible *Computational* Research

1. For every result, keep track of how it was produced
2. Avoid manual data manipulation steps
3. Archive the exact versions of all external programs used
4. Version control all custom scripts
5. Record all intermediate results, when possible in standardized formats
6. For analyses that include randomness, note underlying random seeds
7. Always store raw data behind plots
8. Generate hierarchical analysis output, allowing layers of increasing detail to be inspected
9. Connect textual statements to underlying results
10. Provide ~~public~~ access to scripts, runs, and results

From http://dx.doi.org/10.1371/journal.pcbi.1003285

Learn from the mistakes of others

Don’t be “An Astronomer”



OPEN ACCESS Freely available online

PLOS ONE

How Do Astronomers Share Data? Reliability and Persistence of Datasets Linked in AAS Publications and a Qualitative Study of Data Practices among US Astronomers

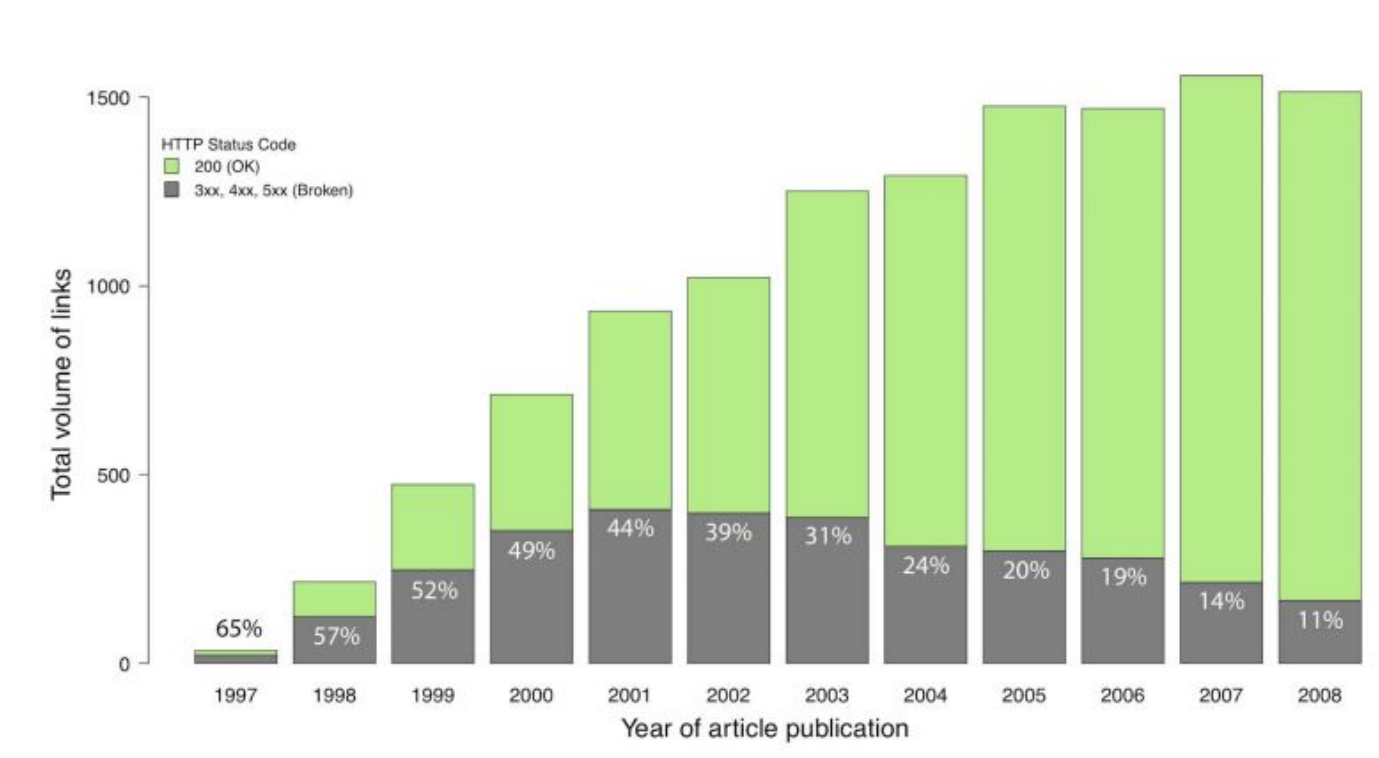
Alberto Pepe[1,2]*, Alyssa Goodman[1,2], August Muench[1], Merce Crosas[2], Christopher Erdmann[1]

1 Harvard-Smithsonian Center for Astrophysics, Cambridge, Massachusetts, United States of America, 2 Institute for Quantitative Social Science, Harvard University, Cambridge, Massachusetts, United States of America

No, I don’t have a website where I store these data. Most of it is in various stages of mess. —An Astronomer

From http://dx.doi.org/10.1371/journal.pone.0104798

Learn from the mistakes of others



From http://dx.doi.org/10.1371/journal.pone.0104798

Learn from the mistakes of others

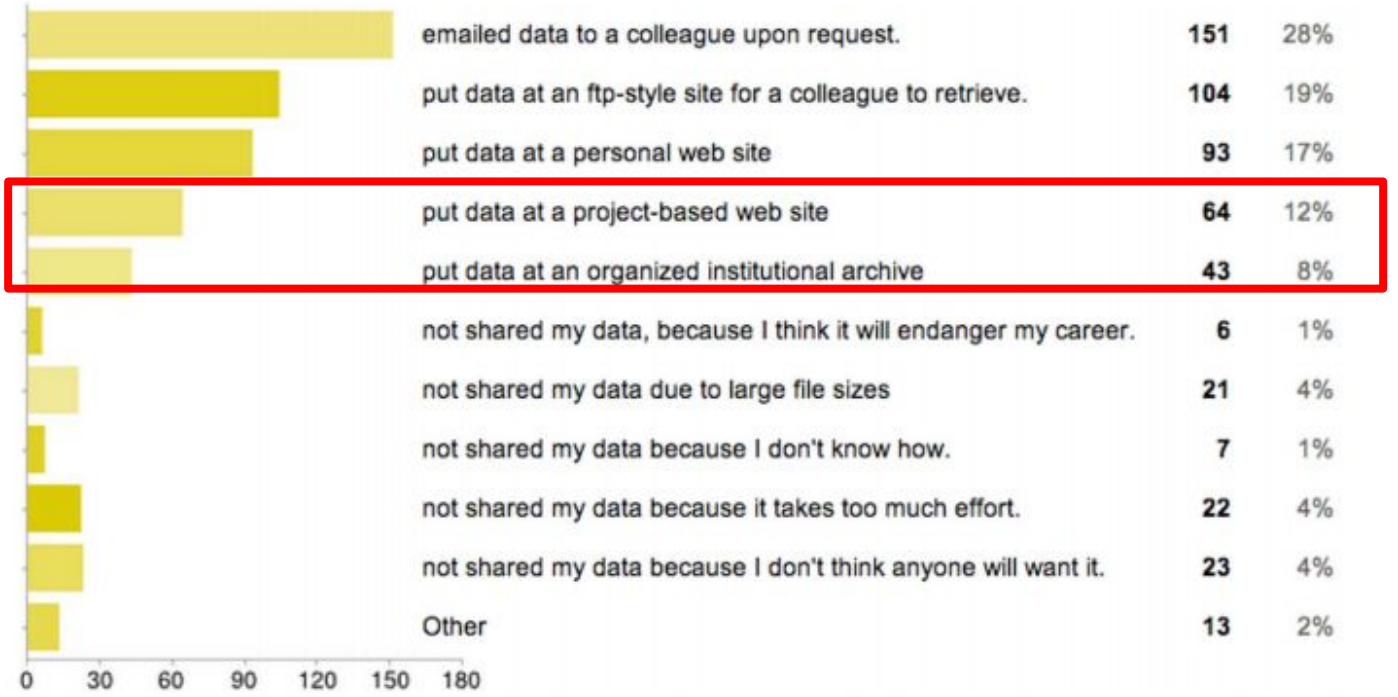


Figure 5. Survey results to question 2: Data sharing practices. Survey results to question: When it comes to sharing DATA you've created, collected or curated, you have?
doi:10.1371/journal.pone.0104798.g005

From http://dx.doi.org/10.1371/journal.pone.0104798

Best Practice

Make reproducibility part of your daily routine

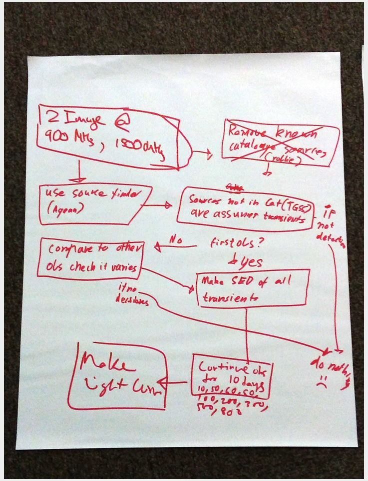
Retrofitting is difficult

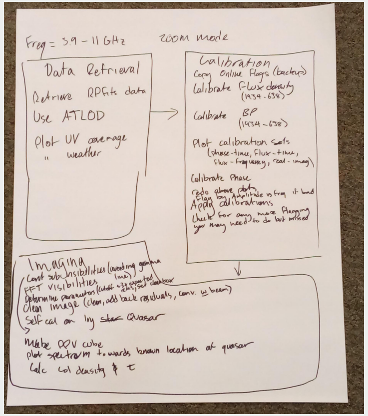
Commit to making your **next** project reproducible from day 1

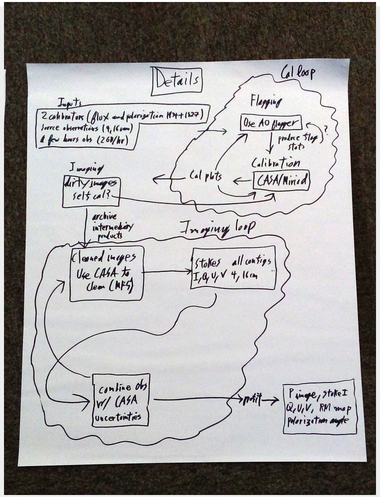
Good automation, documentation, and organisation make reproducibility almost free

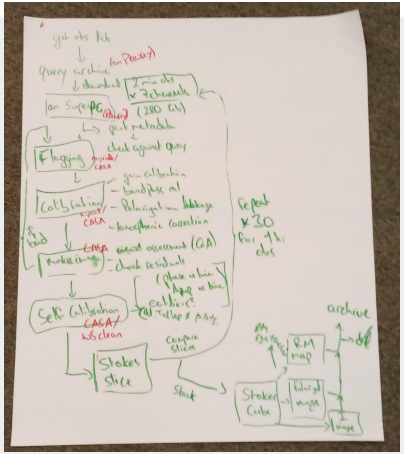
Visualise each task/project as a workflow which needs to be orchestrated with inputs, outputs, and processes.

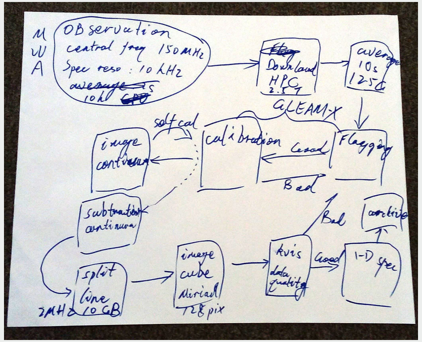
Day 0 Plans

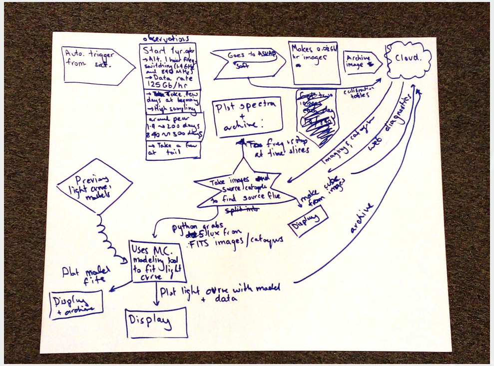

Make SED of all transients
Make light curve


Freq = 3.9 – 11 GHz
zoom mode
Data Retrieval
Retrieve RPFits data
Use ATLOD
Plot UV coverage
Calibration
Imaging

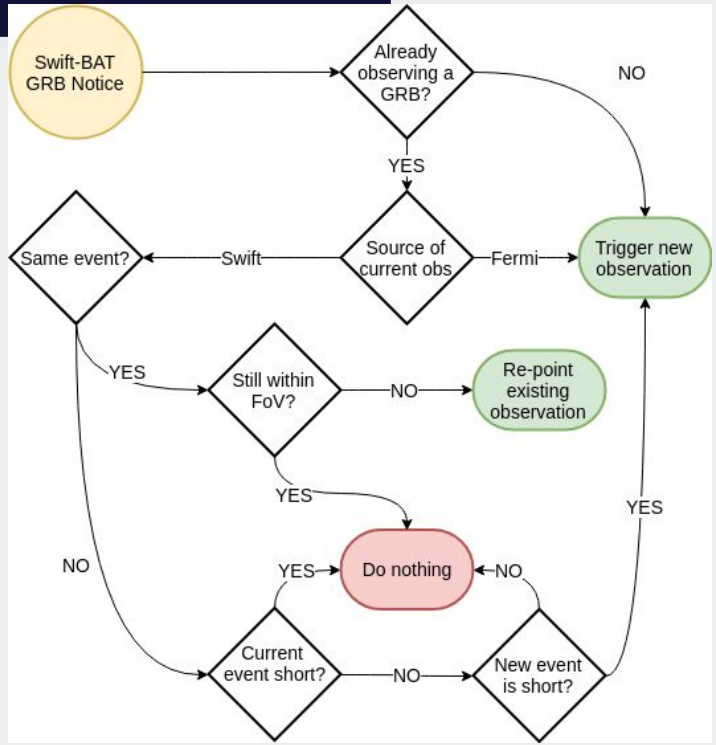

Details
Cal loop
Inputs
Flagging
Calibration
CASA/Miriad
Cal plots
Imaging loop
Cleaned images
Stokes all configs


query archive
Flagging
Calibration
Self Calibration
Stokes slice
Stokes Cube
RM map
archive

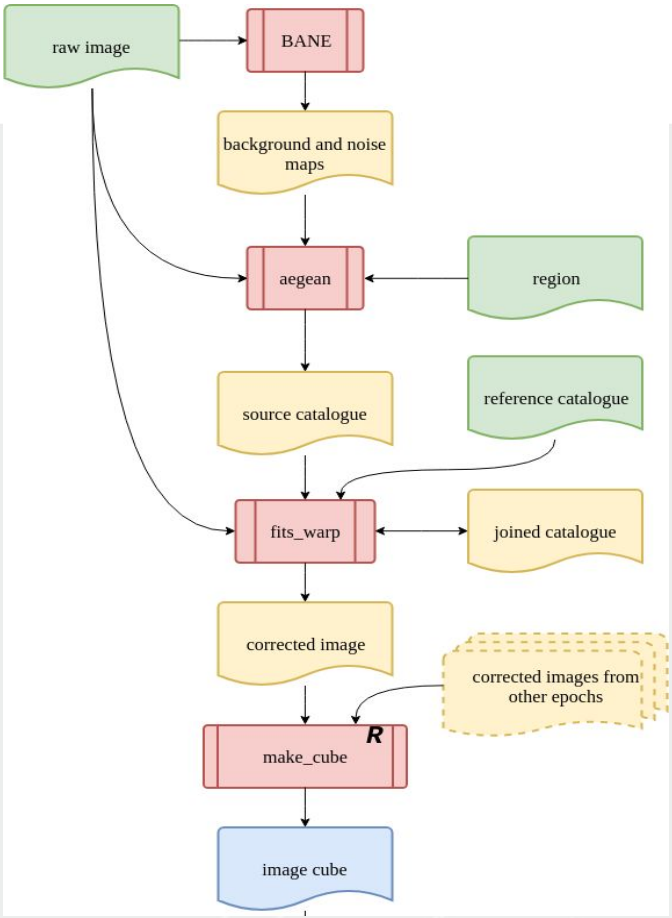

OBservation
central freq 150 MHz
Spec reso : 10 kHz
Download HPC
GLEAM-X
calibration
Flagging
image cube
Miriad


Archive
Cloud.
Plot spectra
archive
Display
Plot light curve with model data

Final implemented workflows



Hancock et al. 2018, 10.1016/j.ascom.2019.02.004



Hancock et al. 2019, 10.1016/j.ascom.2019.02.004

Reproducibility Recap

Making your work/research/reporting/decision making reproducible:

- Builds confidence in results
- Allows trusted methods to be applied to new information
- Saves time and effort by allowing extension/modification

Reproducibility is easy when:

- Workflows are automated
- Metadata is stored
- You embrace it on day 1, not day N-1.

Day 2 – Version Control

The Problem


"FINAL".doc
FINAL.doc!
FINAL_rev.2.doc
FINAL_rev.6.COMMENTS.doc
FINAL_rev.8.comments5.
CORRECTIONS.doc
track changes
JORGE CHAM © 2012
FINAL_rev.18.comments7.
corrections9.MORE.30.doc
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc
WWW.PHDCOMICS.COM

Why Version Control?

As data scientist, we spend much of our time writing code, whether it be for data cleaning, machine learning, or visualisation. As such, our codes are often constantly evolving. By putting all of our code under version control we can:

- **tag code** versions for later reference *(via tags)*.
- record a **unique identifier** for the exact code version used to produce a particular plot or result *(via commit identifiers)*.
- **roll back** our code to previous states *(via checkout)*.
- **identify** when/how **bugs** were introduced *(via diff/blame).*
- **keep multiple versions** of the same code in sync with each other *(via branches/merging).*
- efficiently **share and collaborate** on our codes with others *(via remotes/online hosting).*

Why Version Control?

It's important to also realise that many of the advantages of version control are not limited to just managing code. For example, it can also be useful when writing papers/reports. Here we can use version control to:

- **bring back** that paragraph we accidentally deleted last week.
- **try out a different structure** and simply disregard it if we don't like it.
- **concurrently work on a report** with a collaborator and then **automatically merge** all of our **changes** together.

The upshot is ***you should use version control for almost everything***. The benefits are well worth it...

Introducing Git

Many version control systems exist, we love Git

*Git is a free and open source **distributed** version control system designed to handle everything from small to very large projects with speed and efficiency.*

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows. [Git website]

- *Distributed ->* everyone has their own complete copy of the entire repository and can make changes as they like
- Commiting to a 'central' repository can be done once happy with the changes
- As opposed to *Subversion,* access to the central repo is not required to make changes
- Git is fast (primarily written in C & shell script) and lightweight (as you only track changes)
- Written by Linus Torvalds (creator of Linux)

Introducing Git

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git tracks changes made to files. You can initialise a Git Repository (the .git/ folder inside a project folder) for a project to build a history of your project over time:

- Added files/folders
- Changed files/folders
- Renamed files/folders
- Removed files/folders

Including by who, and when

Why GitHub or GitLab

- ❏ Version Control remotely
- ❏ Visible code and reproducibility
- ❏ Open code and reuse
- ❏ Collaborative code development
- ❏ Open code development

GitHub or GitLab advantages

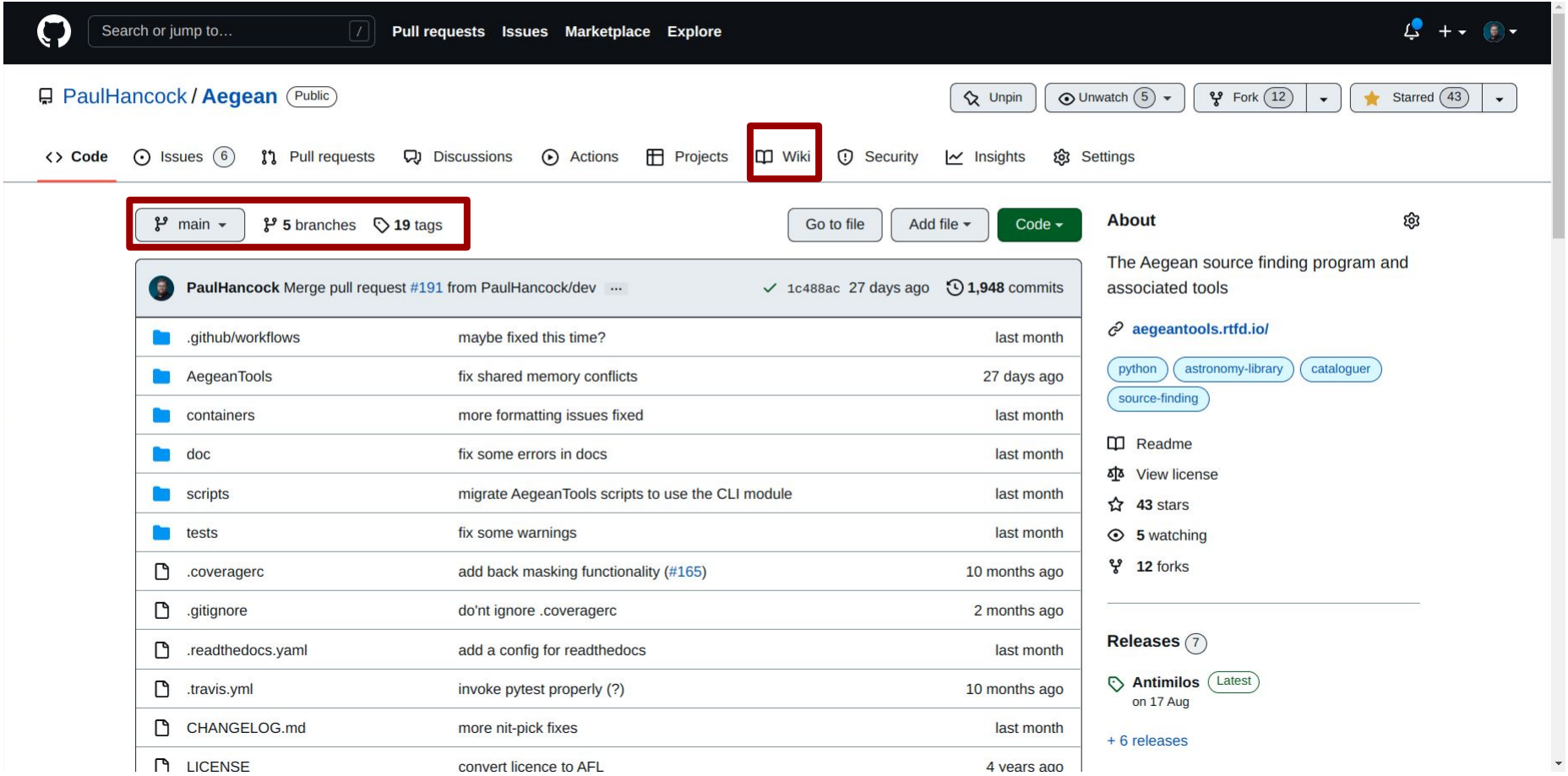
Why we like GitHub/GitLab

- Collaboration tools
- Developer tools
- 3rd party integrations
- Online backup of your work
- Easy/free publish/share work (or keep private)

Example: https://github.com/PaulHancock/Aegean

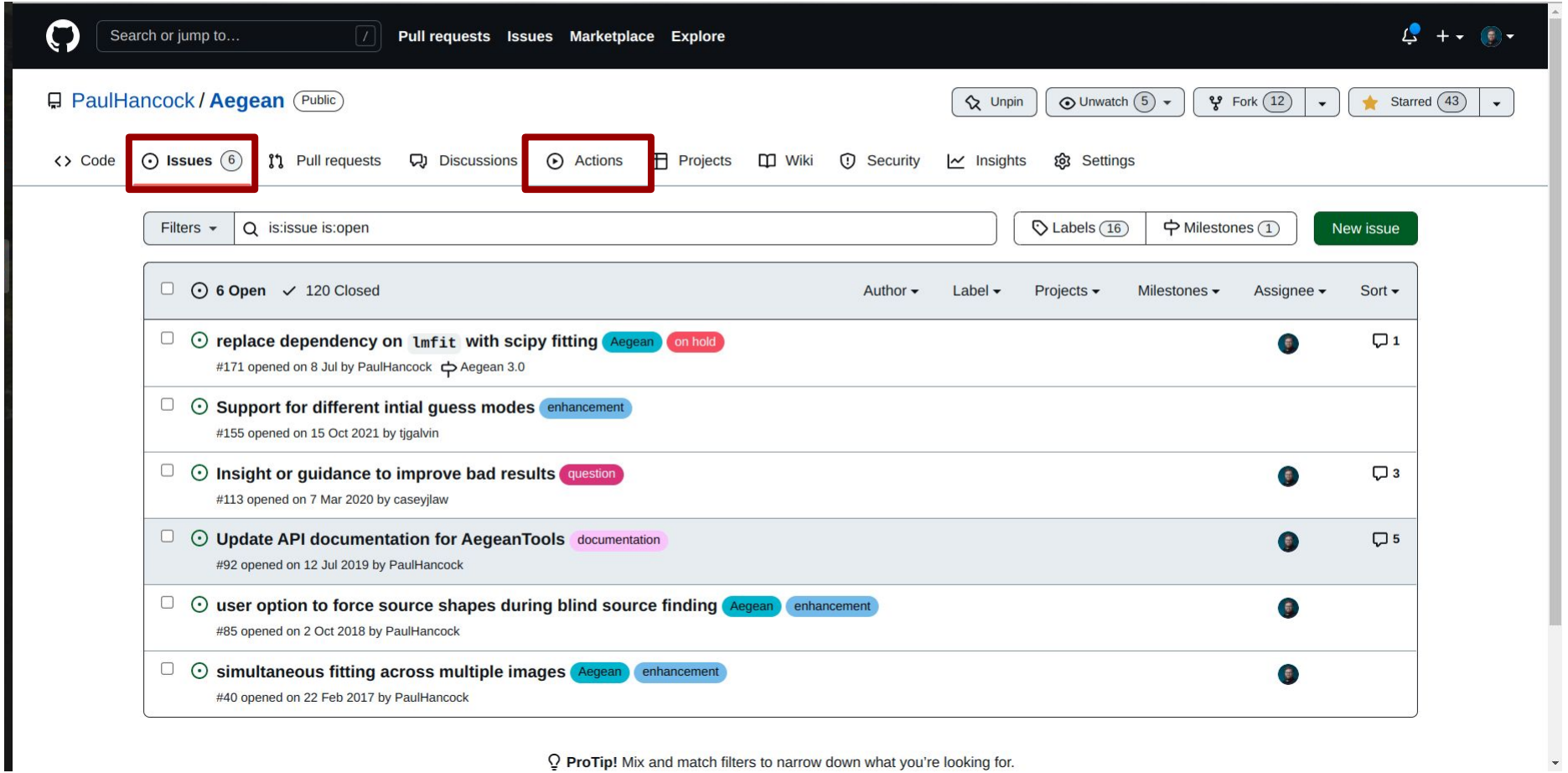
Easy Branch/Tag Management

"Onsite" Documentation



Issue tracking and collaborative tools

Automated tools for testing and deployment



Version Control Recap

Version control helps:

- Identify when/where issues are introduced
- Facilitate safe testing of new ideas through branches
- Track milestones/versions through tagging
- Collaboratively work with *fewer* conflicts

Online Git repositories such as GitHub/GitLab add:

- A centralised repo
- Collaborative tools
- Development tools
- An automated deploy/test/document loop via 3rd party plugins
- A natural platform to share/advertise your excellent work

Saving our progress

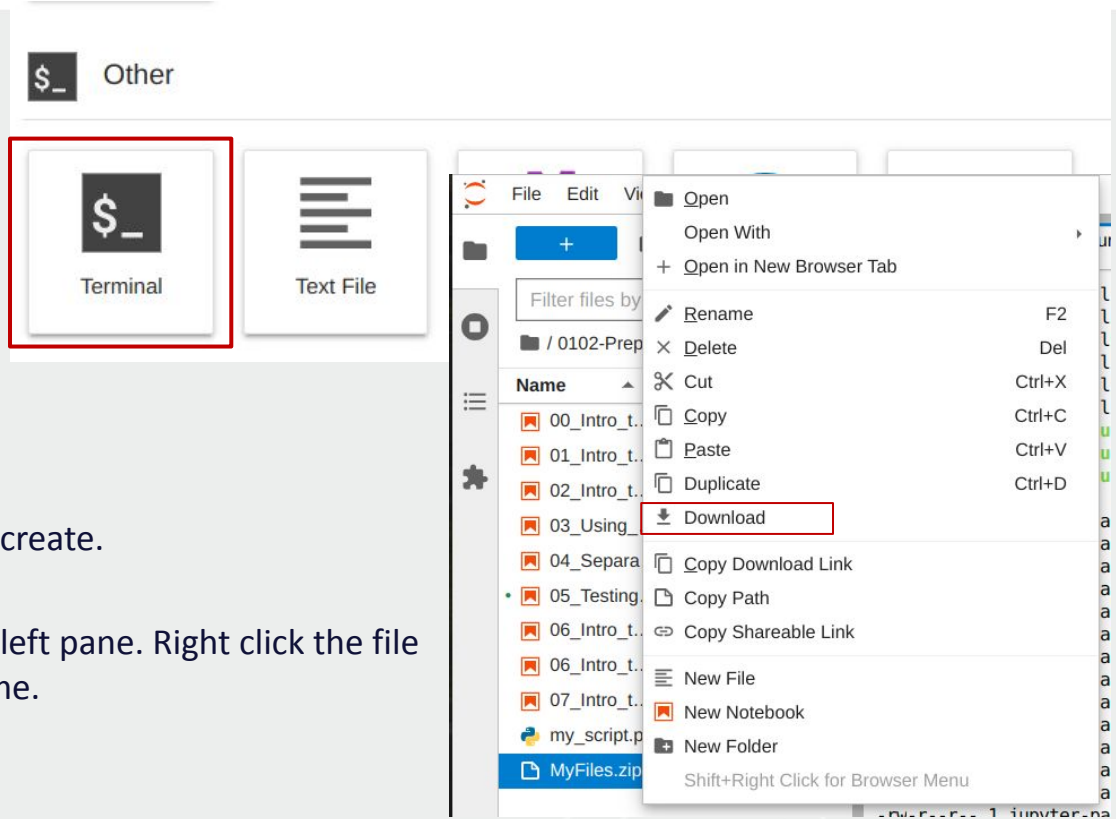
Compress and download our notebooks

Select “terminal” from the launcher tab.

Move to the notebooks directory.

Zip the notebooks using:
> tar -caf MyFiles.zip *
Where MyFiles.zip is the file that we want to create.

You should see the file in the browser on the left pane. Right click the file and select “download” to take your work home.



Day 2 – Reflections

CORE Skills

A CORE partnership

COREHUB.COM.AU/**SKILLS**