



# Delivering Data Science In Resources & Energy

---

## Preparatory: Introduction to the tools and concepts

### Day 1 & 2

Program tools & participant setup

---

Dr Paul Hancock, Dr Dan Marrable

**Curtin Institute for Computation**

The material in this tutorial is inspired by & adapted from the [ADACS good code etiquette workshop](#), the Software Carpentry lesson on [version control](#), as well as <https://guereslib.github.io/Reproducible-Research-Things/>



# Program Timeline

15 Day Professional Program																								
1 Day Leader Engage		15 Day Professional Program																						
1 day Leading Data Scientists Executive Program		Data Science Springboard Preparatory		Introduction to Data Projects		Data Analysis		Data & Communication Sandbox		Data Fusion and Machine Learning		Data Fusion Sandbox		Special Data Types - Time-series & Networks		Special Data Types - Knowledge Discovery from Natural Language Processing & Text Mining		Spacial Data		Capstone Project Development & Presentation		Capstone Propeller		
1/2-1 Day*	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17	Day 18	Day 19	Day 20	Day 21	Day 22	Day 23	
September 2021	Wed 15 Sept 2021	Thurs 16 Sept 2021	Wed 22 Sept 2021	Wed 29 Sept 2021	Wed 6 Oct 2021	Wed 13 Oct 2021	Wed 20 Oct 2021	Wed 27 Oct 2021	Wed 3 Nov 2021	Wed 10 Nov 2021	Wed 17 Nov 2021	Wed 24 Nov 2021	Wed 1 Dec 2021	Wed 8 Dec 2021	Wed 9 Mar 2022									
Enabling your people's data science capability in context of the 15 day program	Introduction to the Program Tools: The Why?	Introduction to the Program Tools: Where does Data Science come from?	Zero to Data Science in a day	Getting to know the Program Tools: Data munging and exploratory data analysis	Simple predictions: Regression and statistical model building	Multivariate analysis and model building	Effective data storytelling: Communicating results to non-technical audiences	Pros and cons of commonly used statistical and machine learning techniques I	Pros and cons of commonly used statistical and machine learning techniques II	Consolidate approaches covered and test on datasets	The 4th dimension and predictions	Finding needles in wordstacks	Spatial analytics and predictions	Capstone Project pitches to leadership	Project Review									

## Leading Data Scientists (Executive Program)

To connect leadership thinking to an enabling organisation-wide data science culture, by supporting the 15 Day Data Science Springboard participants' Capstone Project selection, aligning with strategic priorities and ensuring a deployment pathway into the business.

## Data Science Springboard (Professionals Program)

To enhance data science competency of technical, data-centric staff who can execute the organisation's data science strategy, by (a) transforming data into actionable outcomes, (b) evaluating which tool to use, why and when, (c) appreciating good practice in data science, and (d) understanding how to work with data scientists.



# Outline

1. Overview of the tools used (Anaconda, Python, Github and Git)
2. Intro to Jupyter Notebooks
3. Navigating your computer
  - a. File structure
  - b. Using jupyter notebooks and Python to move around your filesystem
- > Intro to Jupyter
- > Intro to Navigation
4. Project set up
  - a. Package management and Environments
5. Where to find help
- > Intro to Python
6. Good code etiquette + example notebooks
- > Intro to Pandas
- > Intro to Automation
7. Reproducibility
8. Version control

Day 1

Day 2

The background features a complex, glowing purple network of interconnected lines and dots, resembling a digital or biological structure. A large, semi-transparent white circle is positioned on the right side. In the top left corner, there's a white curved line and a small white dotted line. The overall aesthetic is futuristic and scientific.

# Day 2



# Outline

1. Overview of the tools used (Anaconda, Python, Github and Git)
2. Intro to Jupyter Notebooks
3. Navigating your computer
  - a. File structure
  - b. Using jupyter notebooks and Python to move around your filesystem
- > Intro to Jupyter
- > Intro to Navigation
4. Project set up
  - a. Package management and Environments
5. Where to find help
- > Intro to Python
6. Good code etiquette + example notebooks
- > Intro to Pandas
- > Intro to Automation
7. Reproducibility
8. Version control

Day 1

Day 2

# Intro to Pandas

[Intro\\_to\\_pandas.ipynb](#)

# Intro to Automation

[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

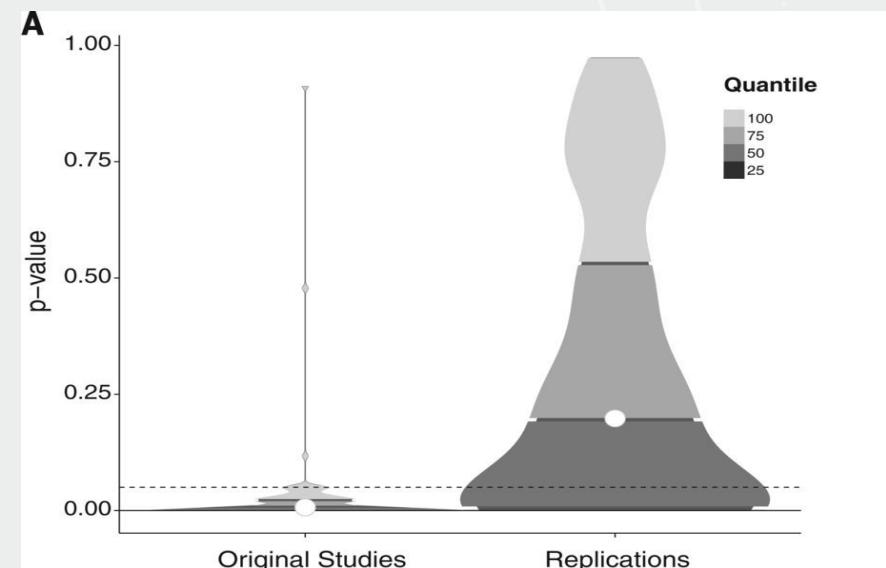
### Estimating the reproducibility of psychological science

Open Science Collaboration<sup>\*†</sup>

+ Author Affiliations

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

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





# 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.

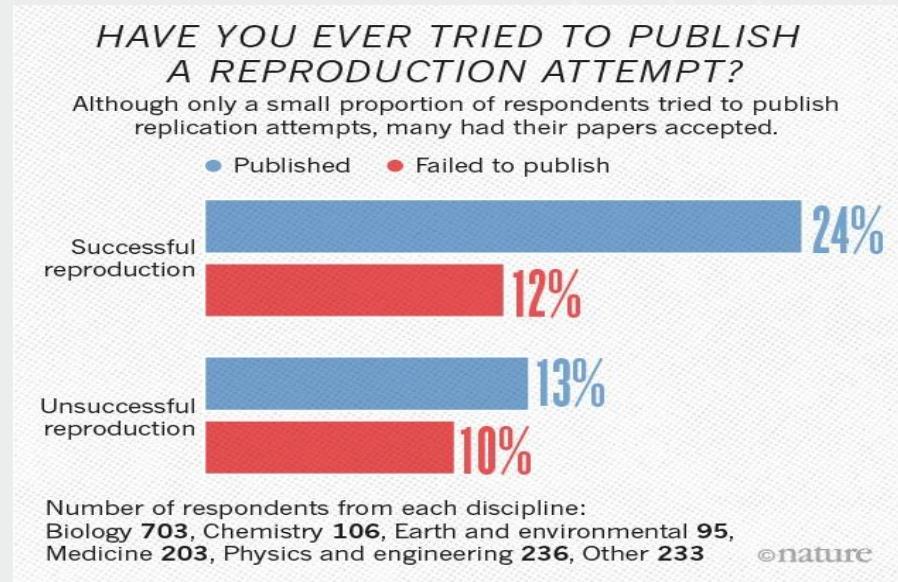
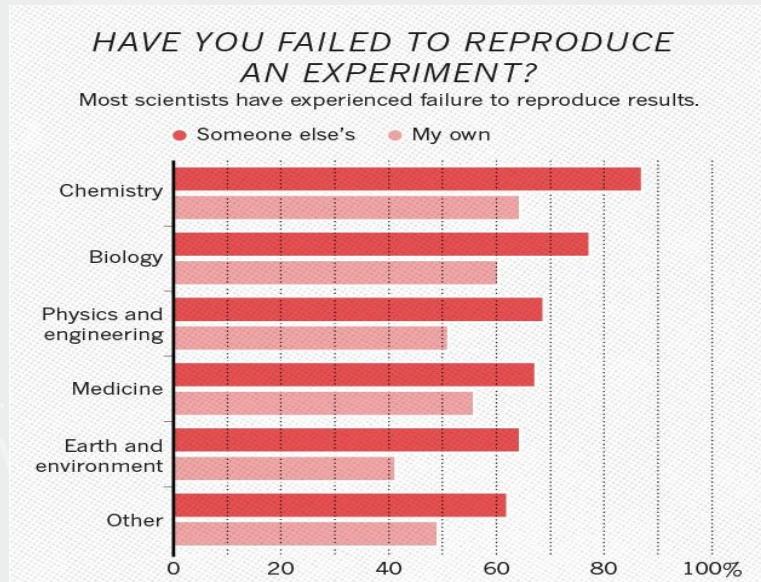
Adaptation allows you to reuse methods that you are confident with.

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



# We should all be worried about reproducibility

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





## 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?

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



## 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?

- Full automation
- Human readable recipe
- Scripted process
- 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

5:00  
Stop

**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.

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



# 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
- Bus 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: / \ " ' \* ; - ? [ ] ( ) ~ ! \$ { } &lt; # @ & | **space** tab newline



# Discussion

5:00  
Stop

## 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 \*v2.png
6. fig02\_core\_instructions\_presentation.png



# 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



# 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



# 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



# 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



# 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



# 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



# Learn from the mistakes of others

Don't be "An Astronomer"

OPEN ACCESS Freely available online

PLOS ONE

CrossMark click for updates

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

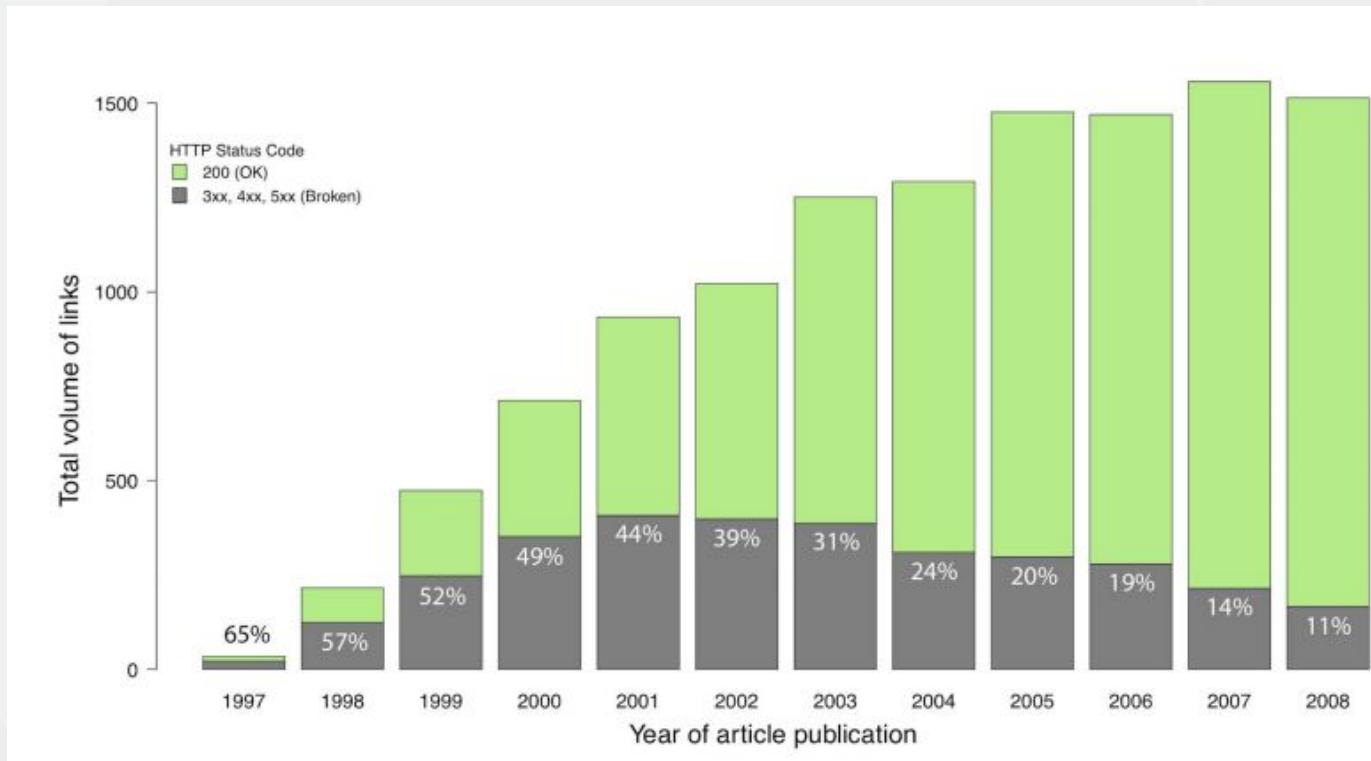
Alberto Pepe<sup>1,2\*</sup>, Alyssa Goodman<sup>1,2</sup>, August Muench<sup>1</sup>, Merce Crosas<sup>2</sup>, Christopher Erdmann<sup>1</sup>

<sup>1</sup> Harvard-Smithsonian Center for Astrophysics, Cambridge, Massachusetts, United States of America, <sup>2</sup> 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

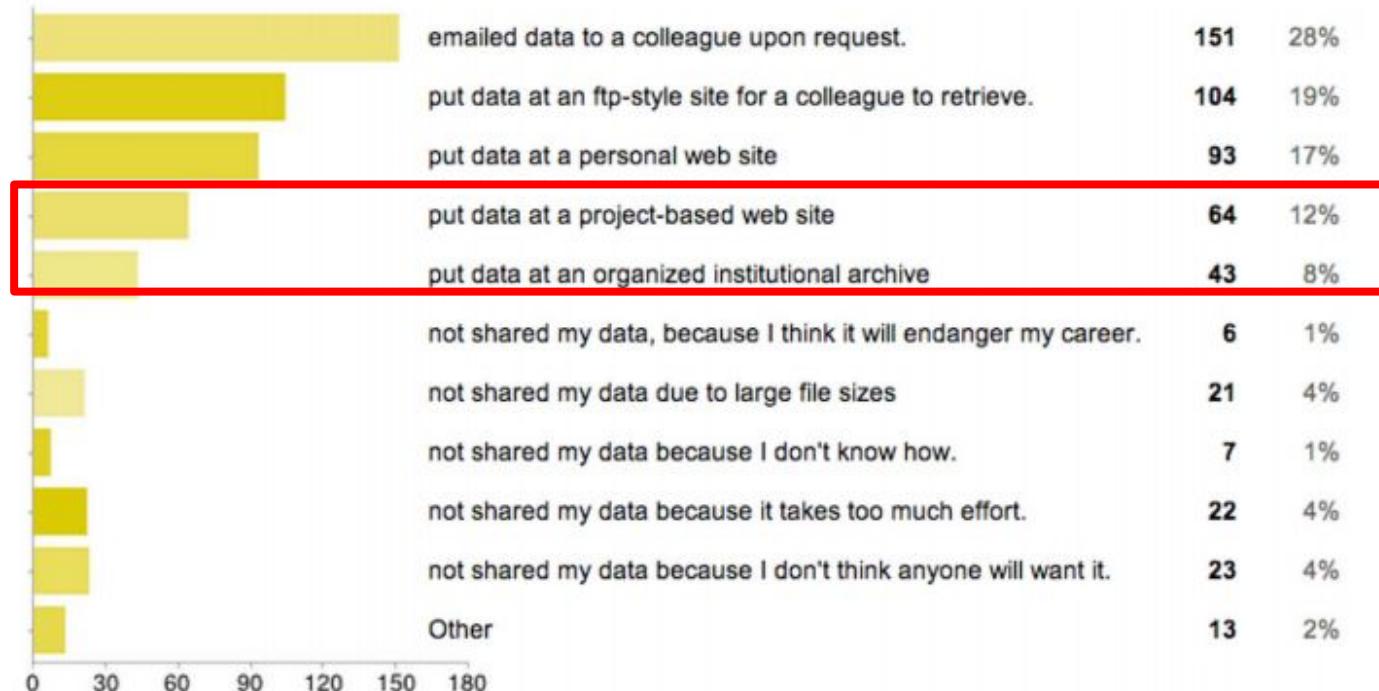


## Learn from the mistakes of others





## 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



## Best Practice

### Make reproducibility part of your daily routine

Retrofitting is difficult

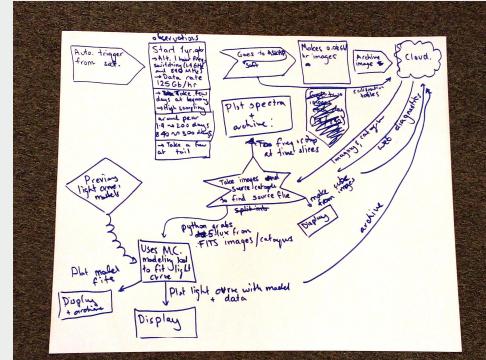
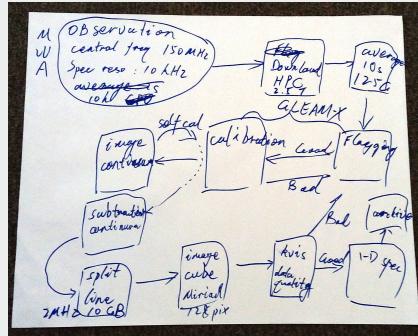
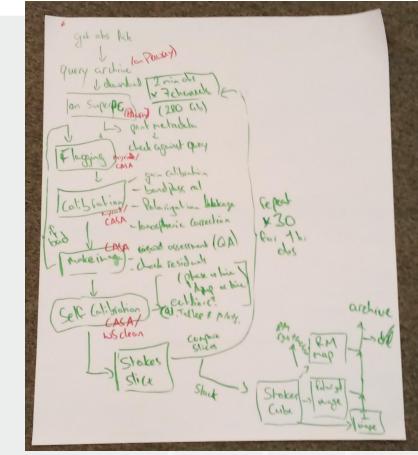
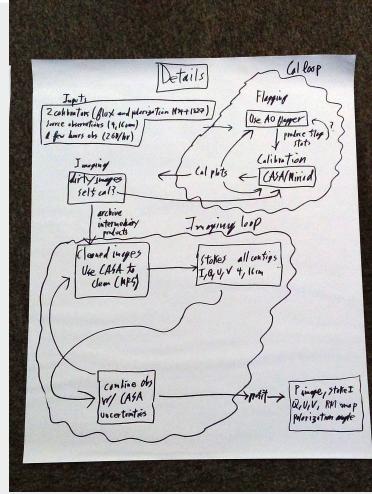
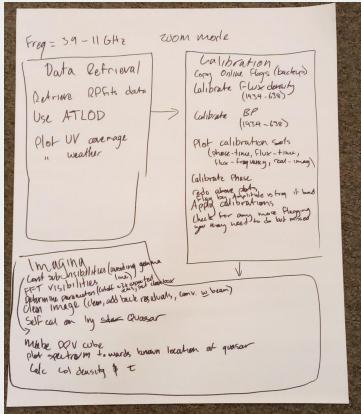
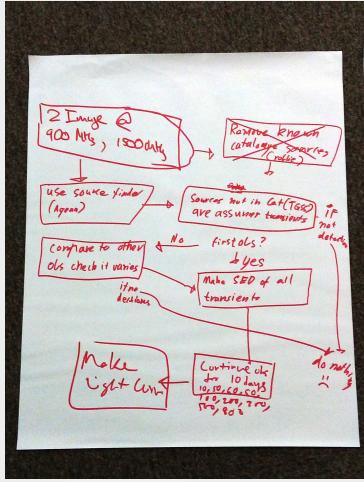
Commit to making your **next** project reproducible from day 1

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.

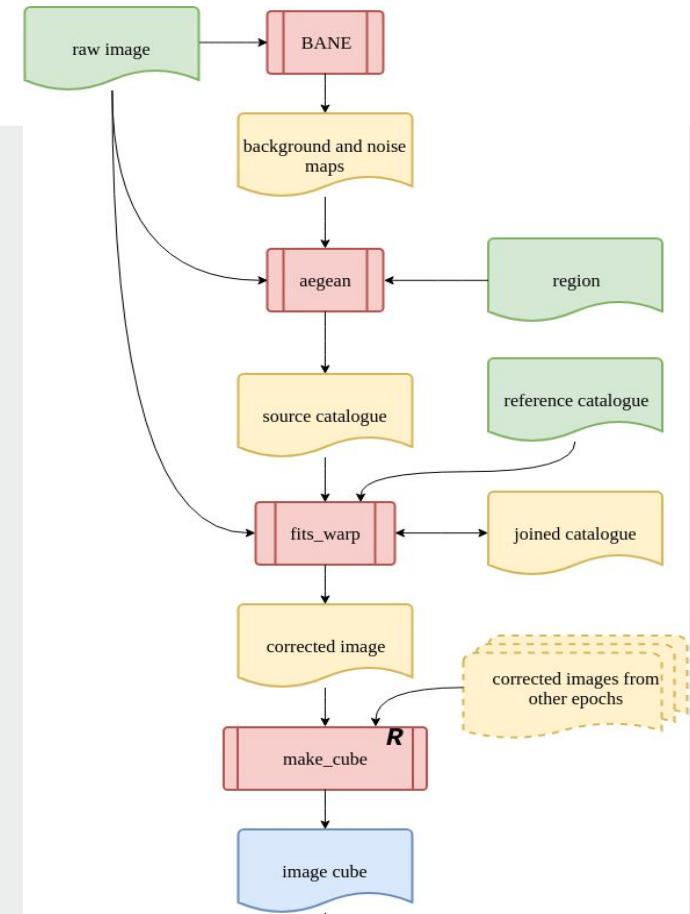
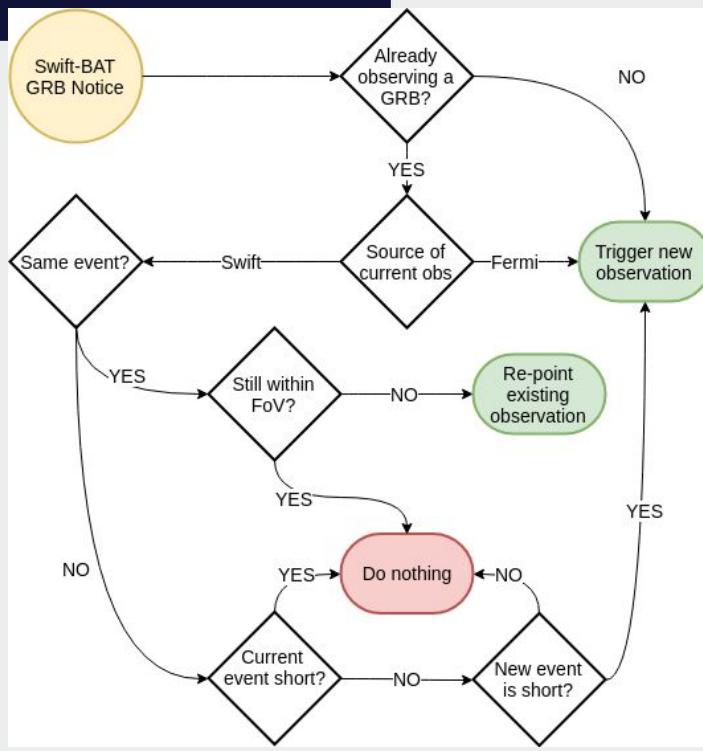


# Day 1 Plans





# Final implemented workflows



# 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

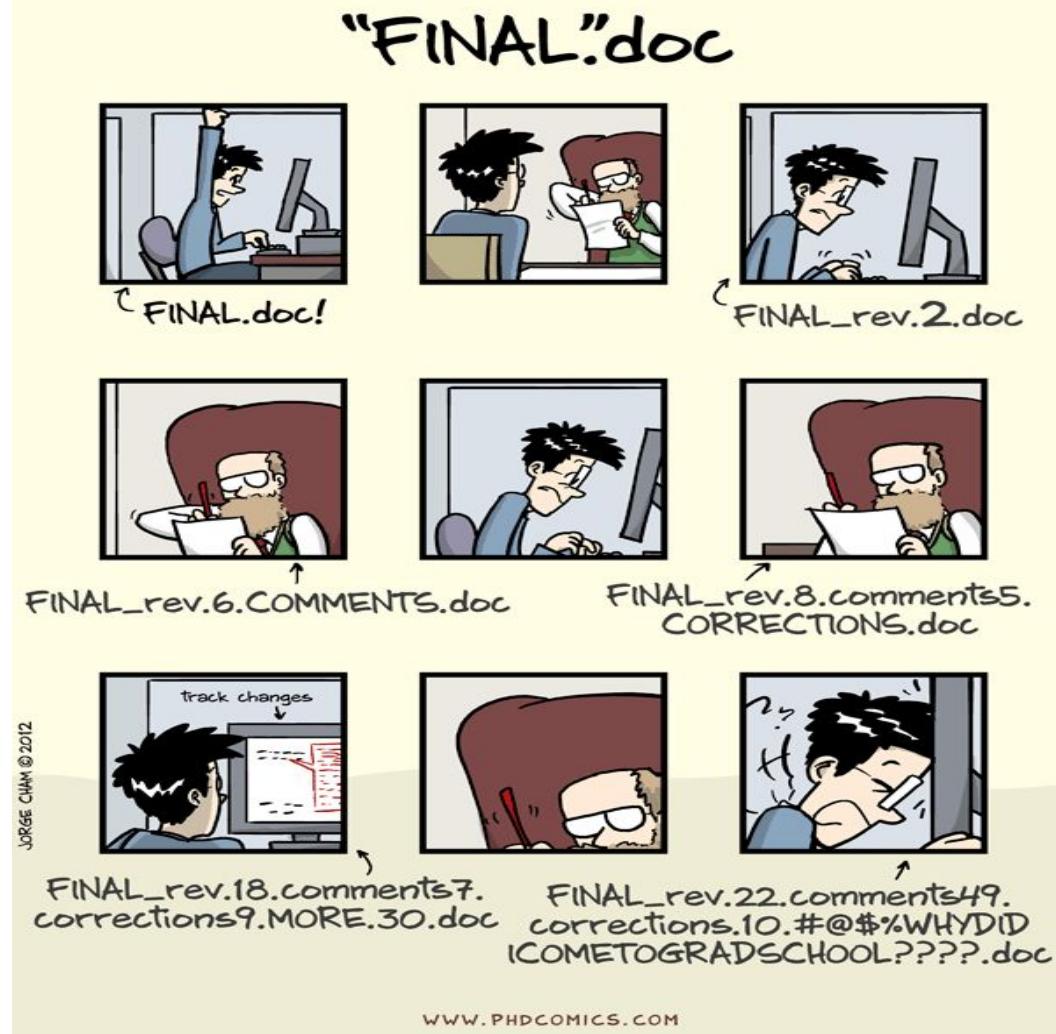
Reproducibility is easy when:

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

# Day 2 – Version Control



## The Problem





# 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 paper** 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
- Committing to a ‘central’ repository can be done once happy with the changes
- As opposed to *Subrversion*, 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

# 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>

**Issue tracker**

**Easier Branch/Tag Management**

**“Onsite” Documentation**

**Code** Issues 17 Pull requests 1 Actions Projects Wiki Security Insights Settings

master 2 branches 16 tags Go to file Add file Code

PaulHancock ensure that masking is applied as intended ... 169ecf3 on 5 Aug 1,718 commits

AegeanTools ensure that masking is applied as intended 3 months ago

doc add reqs for RTD 3 years ago

scripts update date/version 3 months ago

tests create test to catch issue #129 3 months ago

.coveragerc add tests with multiple cores 3 years ago

.gitignore ignore .pytest directory 11 months ago

.travis.yml test on ubuntu xenial 13 months ago

CHANGELOG.md correct error in calculation of condon errors 3 months ago

LICENSE convert licence to AFL 2 years ago

README.md update zenodo DOI link 13 months ago

setup.cfg allow testing from setup.py 3 years ago

setup.py restrict healpy to <=1.13 versions under python 2.7 3 months ago

**README.md**

## Aegean software tools

Programs included:

- aegean - The Aegean source finding program. All your<sup>^</sup> radio astronomy source finding needs in one spiffy program.
- BANE - The Background and Noise Estimation tool. For providing high quality background and noise images for use with Aegean. at a small fraction of the cost of a full box-car smooth.

**About**

The Aegean source finding program and associated tools

[aegeantools.rtfd.io](#)

astronomy-library source-finding  
python cataloguer

Readme AFL-3.0 License

**Releases** 16

[Aegean-Amorgos](#) [Latest] on 3 Mar + 15 releases

**Packages**

No packages published Publish your first package

**Contributors** 4

PaulHancock Paul Hancock  
jnothman Joel Nothman  
Sunmish Stefan Duchesne  
shaoguangleo Shaoguang Guo(Sk...)

**Collaboration tools**

# Help

Please see the [wiki pages](#) for some help and examples. If you have questions that are not answered in the wiki please feel free to email me. If you have found a bug or some inconsistency in the code please [submit a ticket](#) (which will trigger an email to me) and I'll get right on it.

# Credit

If you use Aegean or any of the AegeanTools for your research please give credit by citing:

- Paper 1, [Hancock et al 2012, MNRAS, 422, 1812](#)
- Paper 2, [Hancock et al 2018, PASA, 35, 11H](#)

Until there is a more appropriate method for crediting software development and maintenance, please also consider including me as a co-author on publications which rely on Aegean or AegeanTools.

# Status



Automated  
build /  
document / QA  
integration

DOI [10.5281/zenodo.3474072](https://doi.org/10.5281/zenodo.3474072)



## 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

# Day 2 – Reflections



**COREHUB.COM.AU/SKILLS**

