

# (Digital) Reproducible Science

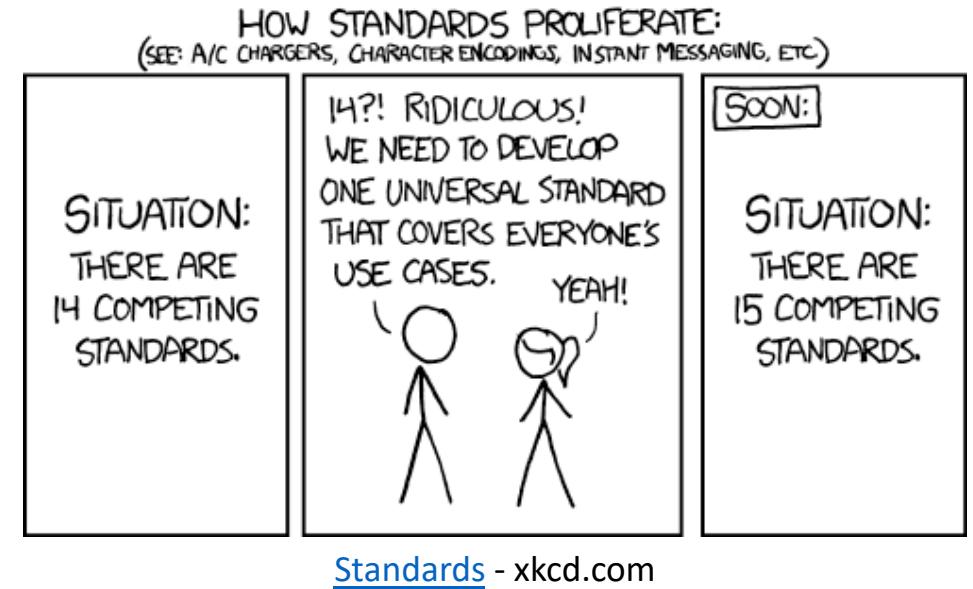
August 2023

Joseph Gum

# Quiz time!

- What is a repository?
  - Differences between code and data repositories?
- Who do you turn in your data/results to?
  - Is your code publicly accessible?
- Name a version control software (VCS)
- What is a branch in git?
- What is a package in Python?

- Have you had trouble running code from others, or sharing your code?
- Is interoperability important to you?
  - Do you feel there are too many standards to understand?



# (Oceanography) Data Stewards Salad

- [ESIP](#)
- [RDA](#)
- [EarthCube](#)
- [FAIR](#)
- [Ocean Best Practices](#)
- [DataONE](#)
- [IEDA](#)
- [Force11](#)
- And so on...



United Nations  
Educational, Scientific and  
Cultural Organization



Intergovernmental  
Oceanographic  
Commission



# An Opinion of Three Cities: Science, Data, and Software

- Science focused:
  - Three “R”s of Science - Repeatability, Replicability, Reproducibility
  - *Reproducibility and Replicability in Science*, 2019
  - The Replication Crisis in Science
  - Solving the Sharing of Software and Data
  - FAIR and CARE principles
  - Data centers/repositories
  - Methods repository - Ocean Best Practices
- Software focused:
  - Community Decisions in Oceanography
  - Working environment: package managers, virtualization
  - Online interactive notebooks: Binder
  - Version control: git, Github
  - Code readability: formatters
  - Code readability part 2: docstrings/comments, autodocs
  - Variable names and standard vocabularies
  - Advanced software engineering
  - Licensing code and data
  - DOIs for code and data

# Reproducible Science: Making Reproducible Experiments Possible

# Three “R”s of Science

- **Repeatability:** **Same** team, **same** experimental setup
  - **Replicability:** **Different** team, **same** experimental setup
  - **Reproducibility:** **Different** team, **different** experimental setup
- 
- **Methods reproducibility:** provide sufficient detail about procedures and data so that the same procedures could be exactly repeated.
  - **Results reproducibility:** obtain the same results from an independent study with procedures as closely matched to the original study as possible.
  - **Inferential reproducibility:** draw the same conclusions from either an independent replication of a study or a reanalysis of the original study.

# *Reproducibility and Replicability in Science, 2019*

- Recently released report from the National Academies of Sciences, Engineering, and Medicine, commissioned by Congress/NSF
- “prompted by concerns about the reproducibility and replicability of scientific research”
- Recommendations on what scientists, funders, journals/conferences, institutions should do in the future
- Recommendations:
  - 6-3: Funding open-source, useable tools and infrastructure for reproducibility
  - 6-5: NSF recommendations related to repositories
  - 6-7: Journals and societies set progressive 3R policies for submissions

# 2023: The Year of Open Science



- White House declared 2023 as “the Year of Open Science”
- Initiatives across all federal agencies to promote Open Science
- NASA – TOPS program, amazing public resource about what this all means
- 2022 – White House Office of Science and Technology Policy issued the [Nelson Memo](#) guiding what agencies must do in future
  - Tighter requirements for all of science to publish open access sooner

# LK-99 – A room temperature superconductor?

- Not up to speed? [Here's a decent overview](#)
- This is exciting! But why is it so hard to replicate?
  - Multiple labs and private individuals around the world are attempting to recreate the material that the authors have a hard time making reliably



Hyun-Tak Kim (CC BY 4.0)

# The Replication Crisis in Science

- Not just social and life sciences!
- [Resplandy et al., 2018](#) – “Quantification of ocean heat uptake from changes in atmospheric O<sub>2</sub> and CO<sub>2</sub> composition”
  - Data is available, attempting to come up with similar numbers falls short in original paper as noticed by a climate change skeptic
    - A problem of [propagated errors](#)
- As science uses more moving parts, especially code, how can we ease the burden of fulfilling all of these dependencies?
  - Use software to ensure everyone uses the latest scientific conclusions?
  - Automatic updating of libraries?

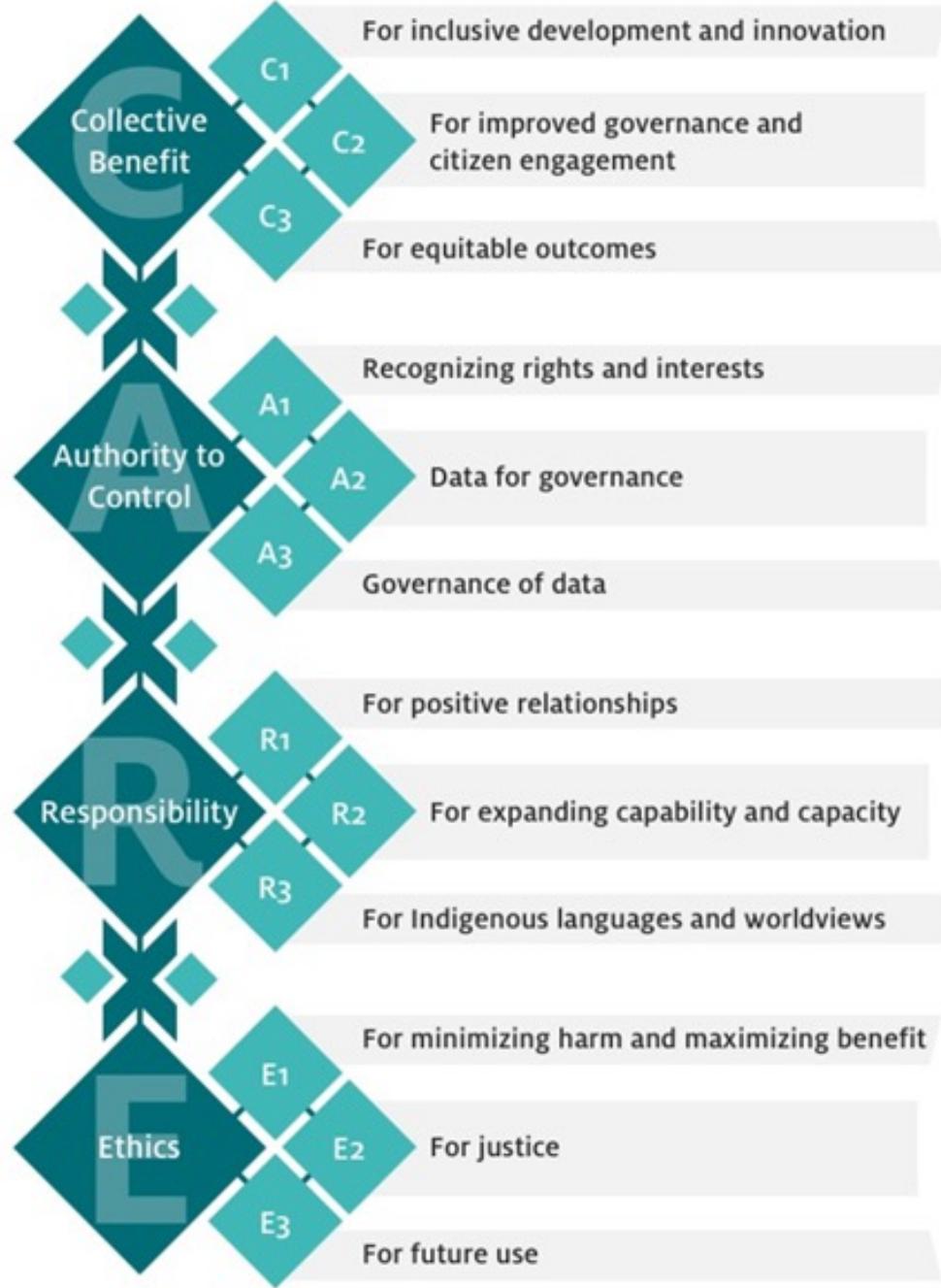
# Solving the Sharing of Software and Data

- What level of replicability is needed for software and data?
  - Loose code – search for a researcher's personal website, hope the links are current and code installs neatly with minimal emails/forum searches?
  - Packaging –package manager, data centers?
  - System image – just boot up an image that contains everything you need?
- Are current scientific best practices being followed?
  - Are the best science practices of the day being followed? Or are outdated libraries being used with ad-hoc fixes to make them current?
- What is the minimum level of fuss a user should expect to encounter?
  - Language expert? No experience required?

# FAIR Principles



- Make data easier to use: “guidelines to improve the findability, accessibility, interoperability, and reuse of digital assets”
- **What does FAIR mean?**
  - **Findable**
  - **Accessible**
  - **Interoperable**
  - **Reusable**
- “Is your data FAIR?” vs “Does your data follow FAIR virtues?”
  - Absolutes versus relatives – start with the relatives

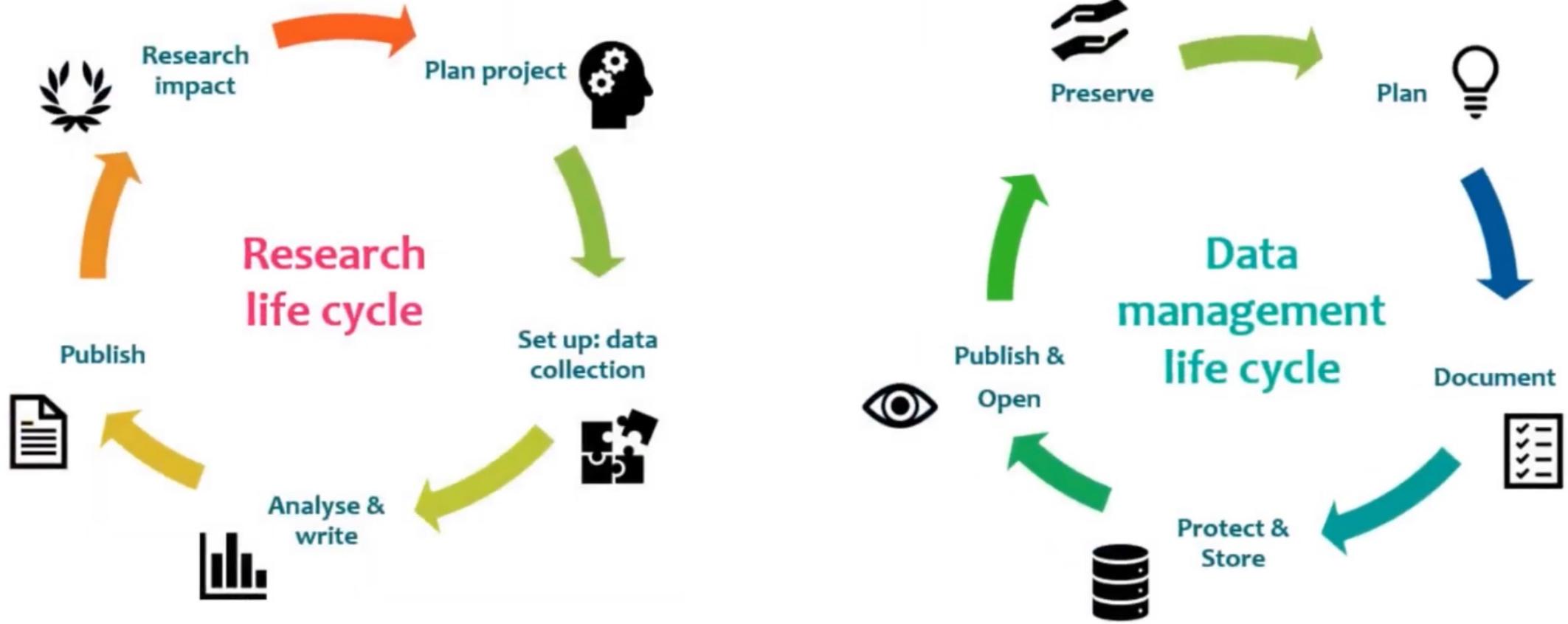


# CARE Principles

- Indigenous Peoples reaction to “data as the new oil”
- A framework for respecting where data comes from
- People and purpose oriented

# Reproducible Data: From Chaos to Results

# What is Scientific Data Management? (Diagram)



...only one of many higher level diagrams on data management!

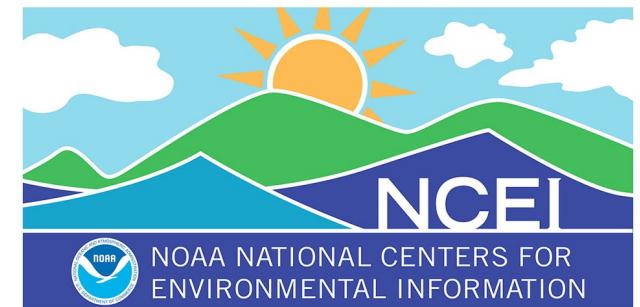
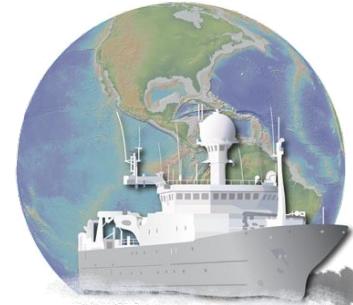
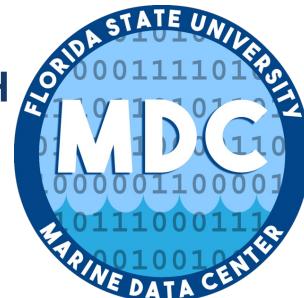
# Chain of Custody of Information

- “Chain of custody” applies even in science!
  - Information collected in the wild or generated in a model
  - Data, metadata that’s been logged, verified
  - Results that’s been published and accessible
  - This includes code, papers, planning documents, etc.
- How your data gets through the chain matters
  - Messy notebooks?
  - Non verified repositories?
  - Predatory journals?
- End goal – trusting what you find, shortening “time to science”

# Data Centers/Repositories



- Data centers are paid to make your data available – talk to us!
  - (It's also a requirement of your grant to submit your data)
- Use our libraries to read our data holdings, or ask us to make them
  - Feel free to contribute too!



# Data Centers/Repositories, Part 2



- Repositories come in different flavors
  - Generalist repositories – will accept anything and everything
    - Ex. [Zenodo](#)
  - Specialized repositories – focus on your field/specialty, supposed to stick around for some time
    - [NSF Ocean Sciences approved repositories](#)
- [CoreTrustSeal](#)
  - A process that verifies repositories meet a minimum standard of requirements to be good holders of your data
  - Is there a repository for presentations on data management/repositories? 🤔

# Repository of Methods – Ocean Best Practices

- DOIs for manuals and handbooks
  - How to do fieldwork, make models, etc.
- UNESCO/IOC project
- Internationally agreed upon methods...
  - ...and publically submitted methods
- Journals: Best Practices in Ocean Observing, Deep Sea Research, etc.



<https://www.oceanbestpractices.org/>

# Reproducible Software: Opinionated Practices with Python

# Why spend effort for reproducibility?

- Faster deployment of code
- Faster understanding of code
- Easier extension of code
- It's good for you and the community!

# How can we measure reproducibility?

- Read more than write – can you read it one month, year from now?
- New installs over current- can you install it one month, year from now?
- Code brittleness – can you extend it one month, year from now?

# Lets see what some people have said about software, data before:

## The ~dangers~ of big data

- Data may be hosted in several different places, which is a waste of resources
- Reproducibility of published analyses:
  - If code isn't readable it is difficult to check
  - Code is often written in proprietary languages
- Data may be used without understanding it or updating it (QC status)
- Where should this data be hosted? Who should maintain it?

PODS-DISCO  
2018, Hawaii



## How should we organize and share our datasets?

### **Access and reproducibility are key!**

Projects like BCO-DMO and Earthcube are creating standards and improving access



Coding standards are important for sharing and reproducibility (IPython/Jupyter notebook/Github)



How can we discover the relationships in large multi-instrument data streams? How can we relate data collected from new technology to old data?

### **How to encourage scientists to participate?**

Requirements by journals and funding bodies  
Building community and tools to keep data sustainable  
Are there non-science funding resources to facilitate this? Creating positions such as "Information Officer"

PODS-DISCO  
2018, Hawaii

# How do we standardize our coding practices?

- “Opinionated” guidelines – [PEP-20](#), “This is the way we are doing things”
- More people following the same opinionated practices – easier to share code!
- Software tools to help write opinionated Python
  - Conda/Conda-forge
  - Black
  - isort (non-notebooks, production code)
- Software tools to help facilitate opinionated Python
  - Git/Github
  - Binder
  - Sphinx (numpy docstrings)
- Science decisions made by communities
  - netCDF-CF
  - Pangeo

# Community Decisions - Oceanography

- Formats: NetCDF (as opposed to [CDF](#))
  - Backends: [HDF5](#), NetCDF3, [zarr](#) (cloud native)
- Formats: ASCII/CSV (old)
  - Easier for initial view, quickly lacking flexibility and metadata
- Standard Vocabularies
  - Regulated vocabularies with strict definitions
  - Propose new names, pass committee approval
  - [CF](#) – Physical parameters ([cf-xarray](#))
  - [BODC](#) lists – Biogeochemical parameters
- Cloud: [Pangeo](#)
  - Standardizing environments to make it easier to run code



# Working environments

- Simplify installing and setting up an environment to run code
- Compartmentalization makes it easier to prevent unforseen mixing that leads to problems
- Package manager:
  - [Conda](#) + [conda-forge](#): easier installation of dependencies, environments
  - Pip: standard Python tool; if all of your packages are in there, fine! (probably not)
- Virtualization:
  - Conda provides virtual environments like `virtualenv`
  - [Docker](#), other tools are good for deployment



# Binder



- A platform for making notebooks runnable online
  - Plug-in to Github
  - Other platforms: Colab, Kaggle
- Allow others to experiment with your code before downloading
- Make DOI notebooks runnable

A screenshot of a GitHub repository page for a "Python environment with requirements.txt". The page shows a list of files: index.ipynb, requirements.txt, and runtime.txt. Below this is a "README.md" file containing the following text:

**Python environment with requirements.txt**

[launch binder](#)

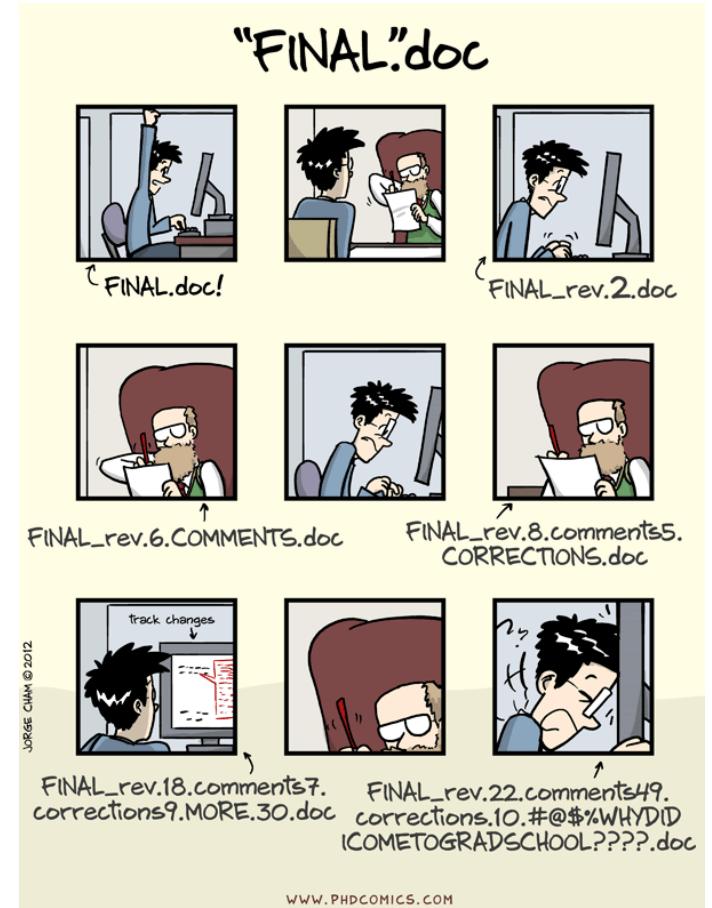
A Binder-compatible repo with a `requirements.txt` file.

Access this Binder at the following URL

<http://mybinder.org/v2/gh/binder-examples/requirements/master>

# Version Control: git, Github

- Easier to keep track of yours/others changes
- Remote copies as failsafes in case of computer failure – but not a replacement for backups
- Commits should be “atomic”
  - Commits are fully formed, not half-done
  - Related edits are grouped into a single commit
  - Commits are small (one change) and often
- Use branches to try new ideas without polluting the master
- Want more [git tutorials?](#)
- [Github](#) – one of many places hosting code repos



"Piled Higher and Deeper" by Jorge Cham  
www.phdcomics.com



# Code Formatting

- [Black](#): “The Uncompromising Code Formatter”
- [isort](#): Sort and trim imports; for production code, not notebooks

```
# in:

def very_important_function(template: str, *variables, file: os.PathLike, engine: str, header: bool = True
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, 'w') as f:
        ...

# out:

def very_important_function(
    template: str,
    *variables,
    file: os.PathLike,
    engine: str,
    header: bool = True,
    debug: bool = False,
):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, "w") as f:
        ...
```

# Code Readability part 2: comments and autodocs

- Docstrings: format for writing documentation in Python
- Flavors: [Numpy](#), [Google](#)
  - Use **numpy** for your projects!
- Document/Autodocs: [Sphinx](#)
  - Automatically generate code documentation
  - Make html, latex, pdf, plain text, etc.
  - Use with [Read the Docs](#) for free hosting
  - Themes: [Read the Docs](#), [Sphinx Book](#)



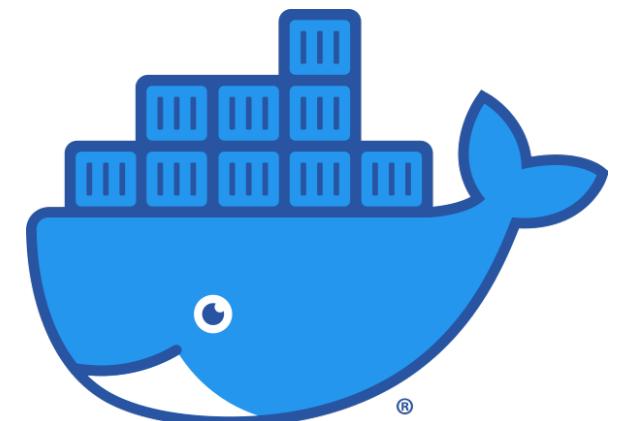
```
class Photo(ndarray):  
    """  
    Array with associated photographic information.  
  
    ...  
  
    Attributes  
    -----  
    exposure : float  
        Exposure in seconds.  
  
    Methods  
    -----  
    colorspace(c='rgb')  
        Represent the photo in the given colorspace.  
    gamma(n=1.0)  
        Change the photo's gamma exposure.  
    """
```

# Code readability – variable names

- Context is important
  - Internal functions vs public API vs data products?
  - Equations implemented in code can make sense to use shorter names
- Variable names
  - Temperature: temperature, temp, tmp, T, [sea water temperature](#), [sea surface skin temperature](#), [TEMPP681](#), [TMESOT01](#)
- Those names in blue are from standard vocabularies
  - Controlled, publically available, names added after proposal process
  - Group Examples: [CF](#), [SeaVox](#), [BODC](#)
- Someone should write a tool that checks files and converts to standard vocabulary names?

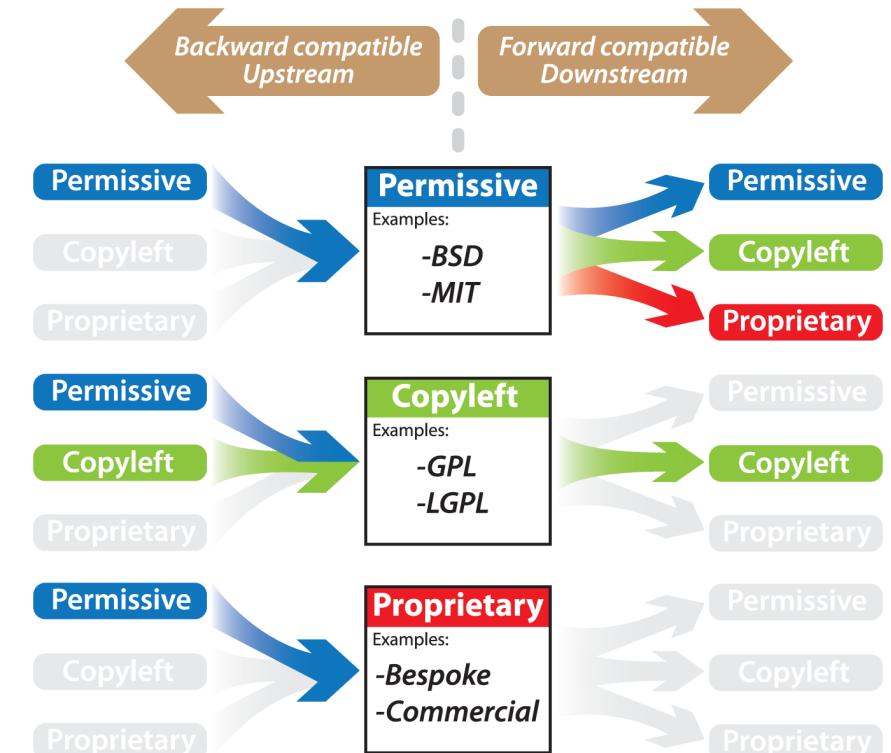
# Advanced Software Engineering

- Python language power features:
  - Decorators, factories, list comprehensions, etc.
- APIs
  - Data: JSON, YAML, XML, REST, SOAP, etc.
- Packaging
  - Version schemes
- Continuous Integration (CI)
  - Special mention: [Pre-commit](#)
- Tests
  - Unit, system/holistic
- Images/Containers
- “Better is the enemy of good”



# Licensing Code and Datasets

- Tool to help you choose:  
<https://choosealicense.com/>
  - Software: MIT, BSD
  - Data: CC
- USA federal government data licenses:  
ODC, CC, GNU (Federal only)
- Publically funded data must be open access, data centers may license holdings
- If no license is stated, a proprietary license is assumed - “all rights reserved”



doi: <https://doi.org/10.1371/journal.pcbi.1002598.g002>

# DOIs for Fun and Profit: Pt. 1



- DOI – Digital Object Identifier, can be made by many organizations
  - Your institution may do so, but you don't need to use them
  - Zenodo is broadly available for all
- Making DOIs for code
  - <https://guides.github.com/activities/citable-code/>
- Making DOIs for data
  - <https://www.zenodo.org/>
  - Your institution may be able to [make DOIs](#)



# DOIs for Fun and Profit: Pt. 2



- Citing DOIs:
  - Zenodo: lets look at [10.5281/zenodo.1256998](https://doi.org/10.5281/zenodo.1256998)
    - Choose citation style
  - Online tool: <https://citation.crosscite.org/>
    - 10.7942/C2008W, "american-geophysical-union", "en-US"
- These tools depend on the DOI fields being filled out properly, so you don't have to manually form them
  - Otherwise, follow recommended style

# Questions?

Want the slides? (DOI coming)

<https://github.com/asx-/reproducible-science>

# Epilogue: Why make Open Source Software?

- Requirements
  - Your funder probably requires publicly available results including data and **software**
  - One common exception: military funded projects
- Collaboration
  - Connect with others using the same software, work together on projects
- Publishing
  - Your code are your methods, especially for computationally intensive projects, and should be auditable by others
- Maintainability
  - As a project grows, someone may possibly maintain your software for free!

# Appendix A: Python Libraries You Might Not Be Using But Should

- Official Standard Libraries
  - [Exceptions](#) – Catch warnings and errors and not fail
  - [Logging](#) – Put logs out to files, different priority levels instead of print statements
  - [Pathlib](#) – Current and easy way of interacting with paths
  - [Pdb](#) – Python debugger
- Community Standard Libraries
  - [Pytest](#) – Testing framework
  - [SciPy](#) Stack – basic tools for scientific computing
  - Plotting – [Cartopy](#), [Holoviews](#), [Matplotlib](#)
  - Cloud Data – [xarray](#), [dask](#), [zarr](#)
  - (Bonus) [NumFOCUS](#) – Foundation that funds a number of science focused Python projects and other software

# Appendix B: Resources for improving (digital) reproducible science

- Carpentries Project
  - Software
  - Data
- OceanHackWeek
  - 2018, 2019

# Appendix C: Arguments for Open Source

- Open Source means open to collaboration, distributed efforts:
  - New opportunities for unknown collaborators to join
  - Distributed costs – instead of one project funding 1 FTE, 4 projects funding 0.3 FTEs
  - Redundancy in maintenance – One person on holiday, retiring, etc. does not bring the entire project down
  - Easier hiring – pay other maintainers to add new features, find other developers working on similar projects able to add to the software, find private industry programmers to help more easily -  
<https://science.mozilla.org/projects>
  - Journals and repositories are moving towards mandated shared code and data – move ahead of the curve early

# Appendix D: Shiny is to R as ??? is to Python?

- Quickly turn Python code into a web dashboard
- Streamlit - <https://streamlit.io/docs/index.html>
- Dash - <https://dash.plot.ly/introduction>