CIML Summer Institute:

Reproducibility in Science and Machine Learning

June 22, 2021

Peter Rose

SDSC

EXPANSE

COMPUTING WITHOUT BOUNDARIES

SAN DIEGO SUPERCOMPUTER CENTER

NSF Award 1928224

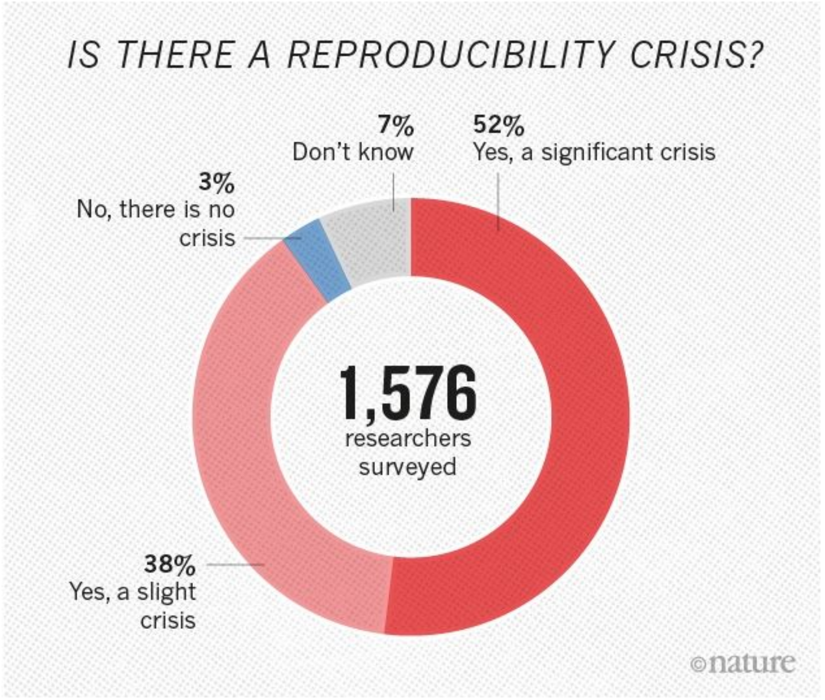
Reproducibility Crisis?

"More than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments."

Nature, 2016, M. Baker, 1,500 scientists lift the lid on reproducibility

"Nature journal editors … will, on a case-by-case basis, ask reviewers to check how well the code works."

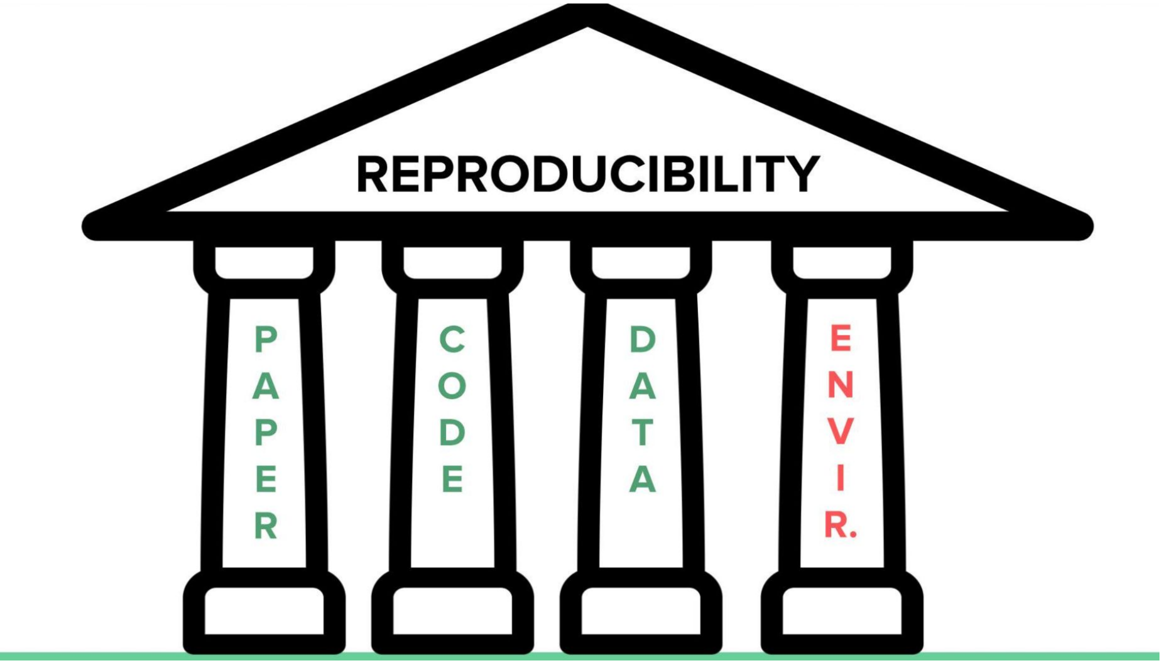
Nature, 2018, Does your code stand up to scrutiny?



Reproducibility*	Reusability	Scalability
obtaining **consistent** results using	obtaining **new** results using	obtaining **new** results using
same input data or parameters	**different** input data or parameters	**large** input data or parameter sets
same computational steps, methods, and code	**same** computational steps, methods, and code	**same** computational steps, methods, and code
same analysis conditions	**same** analysis conditions	**same** analysis conditions

* L. Barba, https://figshare.com/articles/Next_in_Reproducibility_standards_policies_infrastructure_and_human_factors/8194328/1

Four Pillars of Reproducible Research

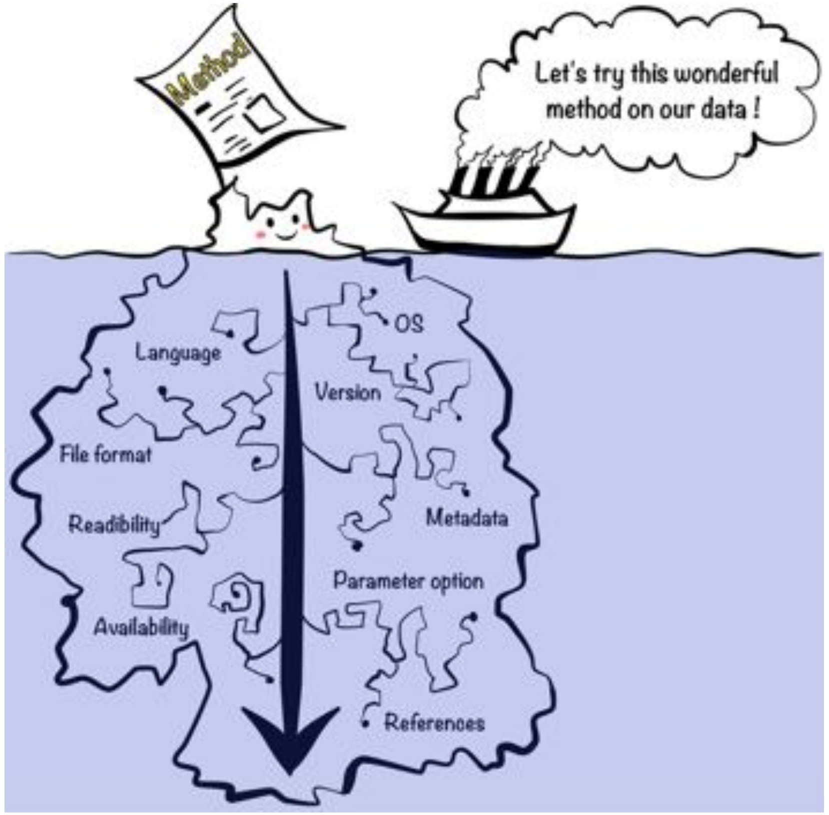


Open Science

- Open access publications
- Open source code
- Open data
- Open execution environment

http://theoryandpractice.org/2016/05/Reproducibility-Symposium/

Barriers to Reproducibility and Reusability



https://doi.org/10.1093/gigascience/giy077

- Missing or incomplete documentation
- Distribution is missing files
- Missing third party package
- Dependencies failed to build
- Runtime error
- Internal compiler error
- My last week:
  - samtools: error while loading shared libraries: libbz2.so.1.0: cannot open shared object file
  - error while loading shared libraries: libz.so.1: failed to map segment from shared object: Operation not permitted
  - /lib64/libc.so.6: version `GLIBC_2.14' not found

S. Turner, bit.ly/madssci-2018-repro

Today you’ll learn techniques and tools to overcome these barriers and publish reproducible workflows and scale up your calculations on HPC!

Tools and Infrastructure

Computational notebooks: combine documentation, code, and results

Scalable compute infrastructure

Open cloud environment to run computational notebooks

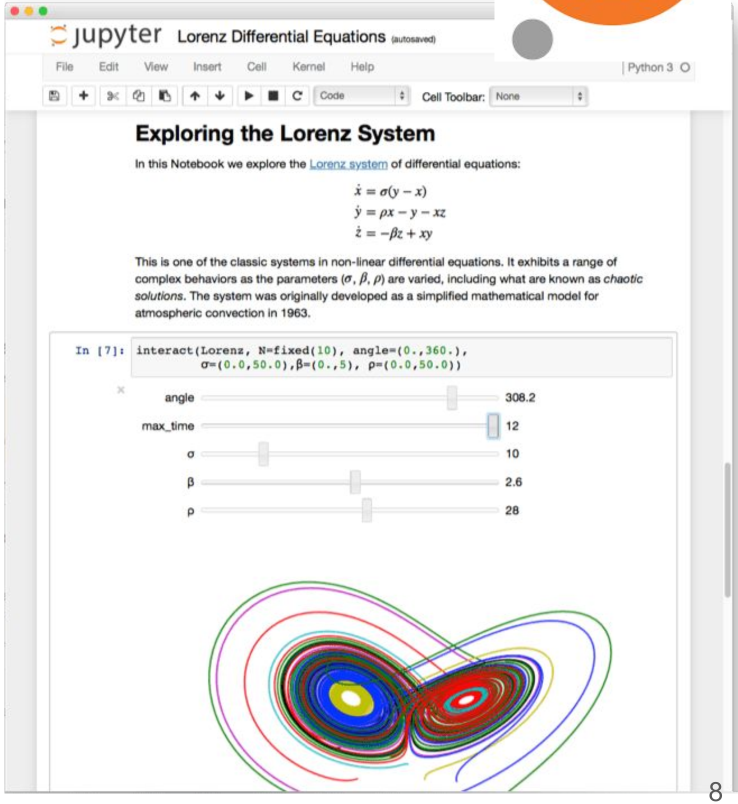
Version-control system for tracking changes in source code

Open-source package and environment management system

Source code repository

Jupyter Notebook

- An open-source web application to create and share documents
- Combines live code, equations, visualizations and narrative text
- Supports over 40 programming languages
- Important tool in support of reproducible workflows
- A document format to save and share computational narratives (.ipynb file)
- > 10 million Jupyter Notebooks on GitHub
- Jupyter Lab, next generation user interface


jupyter Lorenz Differential Equations (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Code
Cell Toolbar: None

Exploring the Lorenz System

In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

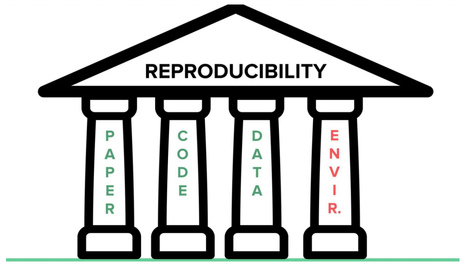
This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

```
angle     308.2
max_time  12
σ         10
β         2.6
ρ         28
```

Setting up your Environment for Reproducibility

Conda



- Package management system
  - Conda installs, runs, and updates open source packages (e.g., NumPy, Pandas) and their dependencies, while checking compatibility with all preexisting packages.
- Environment management system
  - Conda allow you to create, save, load and switch between multiple environments on your local computer, as well as share instructions for how to recreate that environment on a different computer.
- Multi-platform (Windows, MacOS, and Linux)
- Multi-language (Python, R, Ruby, Scala, Java, JavaScript, C/ C++, etc.)

CONDA®

Why Conda Environments?

```
pip install pandas
pip install scikit-learn

Or

conda install pandas
conda install scikit-learn
```

Directly installing packages into your base environment will lead to version conflicts, errors, and non-reproducible results.

```
environment_1

python=3.7
pandas=0.25.0
scikit-learn=0.20.0
```

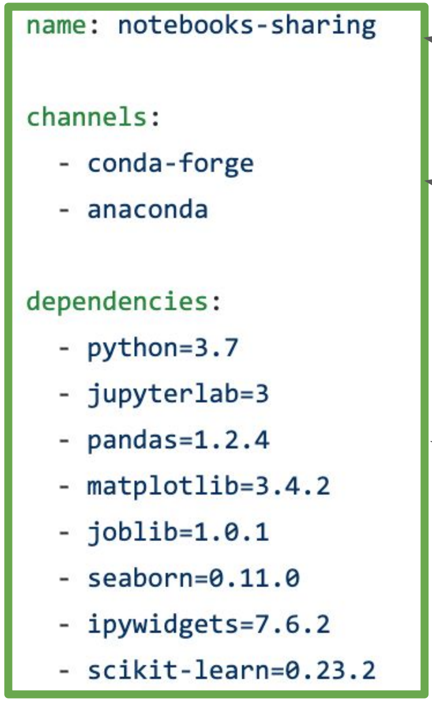
```
environment_2

python=3.9
pandas=1.2.4
scikit-learn=0.24.2
```

By creating conda environments, multiple versions of software packages can co-exists without interference.

Conda environment are portable and can be installed on multiple platforms.

Define a Conda Environments

Create an **environment.yml** file and place it into the top level of your Git Repository



```
name: notebooks-sharing

channels:
  - conda-forge
  - anaconda

dependencies:
  - python=3.7
  - jupyterlab=3
  - pandas=1.2.4
  - matplotlib=3.4.2
  - joblib=1.0.1
  - seaborn=0.11.0
  - ipywidgets=7.6.2
  - scikit-learn=0.23.2
```

Use the same name as your Git repository

Specify the channels where to look for packages.
The order matters!
The conda-forge channel has newer versions than anaconda channel.

To ensure reproducibility and compatibility of the software packages, specify (**“pin”**) the version number.

Try to use the newer or latest versions of software packages to ensure reusability in the future.

https://github.com/sdsc-hpc-training-org/notebooks-sharing/blob/main/environment.yml

Create a Conda Environment

Prerequisite: Miniconda3 (light-weight, preferred) or Anaconda3 installed

See instructions: Expanse (Linux distributions)   Windows 10 or MacOS

Create a conda environment (Expanse uses a network file system. Creating conda environments is **slow!**)

```
conda env create -f environment.yml
```

List your environments

```
conda env list
```

Activate a conda environment

```
conda activate <environment_name>
```

Deactivate conda environment

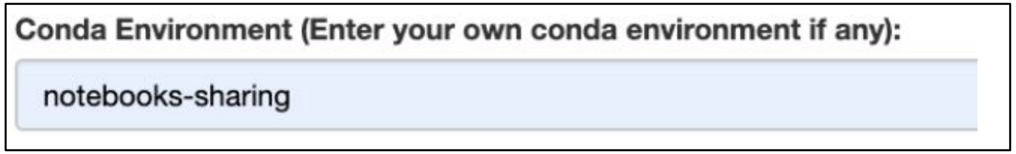
```
conda deactivate
```

Remove a conda environment

```
conda env remove -n <environment_name>
```

Scaling up your Calculations

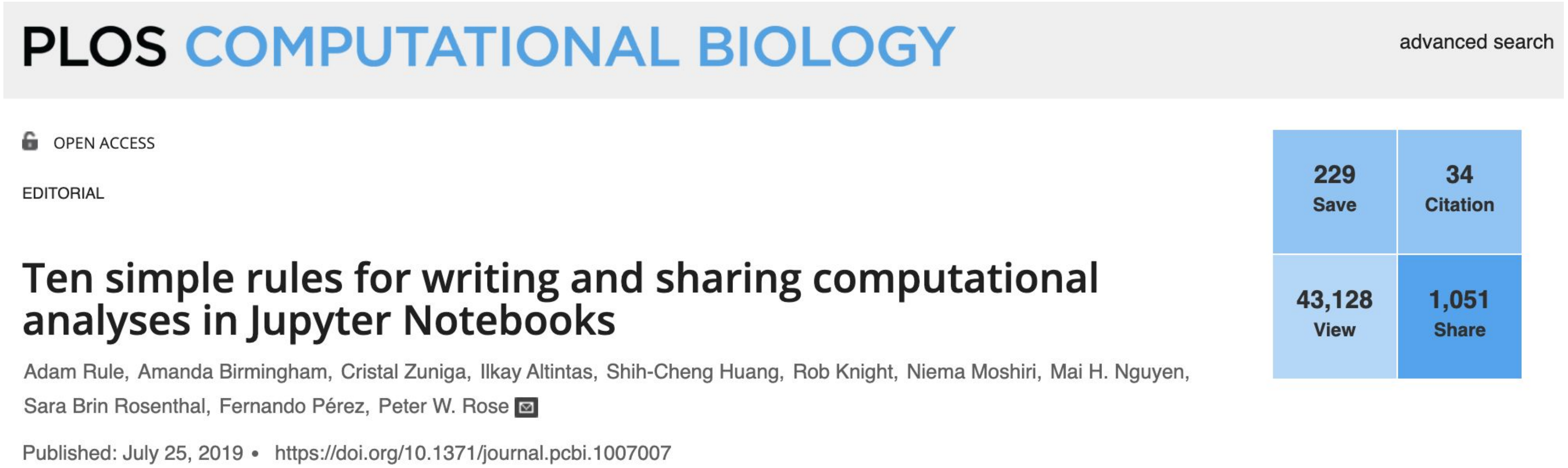
Scaling up from a Laptop to Expanse

1. Create and test conda environment on a laptop or desktop
2. Test your code on a small sample
3. Check code into a Git repository
4. Login to Expanse
5. Clone the Git repository
6. Create and test the conda environment*
7. To run Jupyter Notebook/Lab, specify your environment in:
   a. 
      Conda Environment (Enter your own conda environment if any):
      notebooks-sharing
   b. galyleo script

```
galyleo.sh launch --account sds184 --partition 'shared' --cpus-per-task 1 --memory-per-node 4
--time-limit 00:30:00 --jupyter 'lab' --notebook-dir "/home/${USER}" --conda-env 'notebooks-sharing'
```

* https://github.com/ciml-org/ciml-summer-institute-2021/tree/main/1.4_installing_your_own_miniconda_creating_conda_environments#14_installing_your_own_miniconda_creating_conda_environments

Writing and Sharing Computational Analyses in Jupyter Notebooks


PLOS COMPUTATIONAL BIOLOGY

advanced search

OPEN ACCESS

EDITORIAL

Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks

Adam Rule, Amanda Birmingham, Cristal Zuniga, Ilkay Altintas, Shih-Cheng Huang, Rob Knight, Niema Moshiri, Mai H. Nguyen, Sara Brin Rosenthal, Fernando Pérez, Peter W. Rose

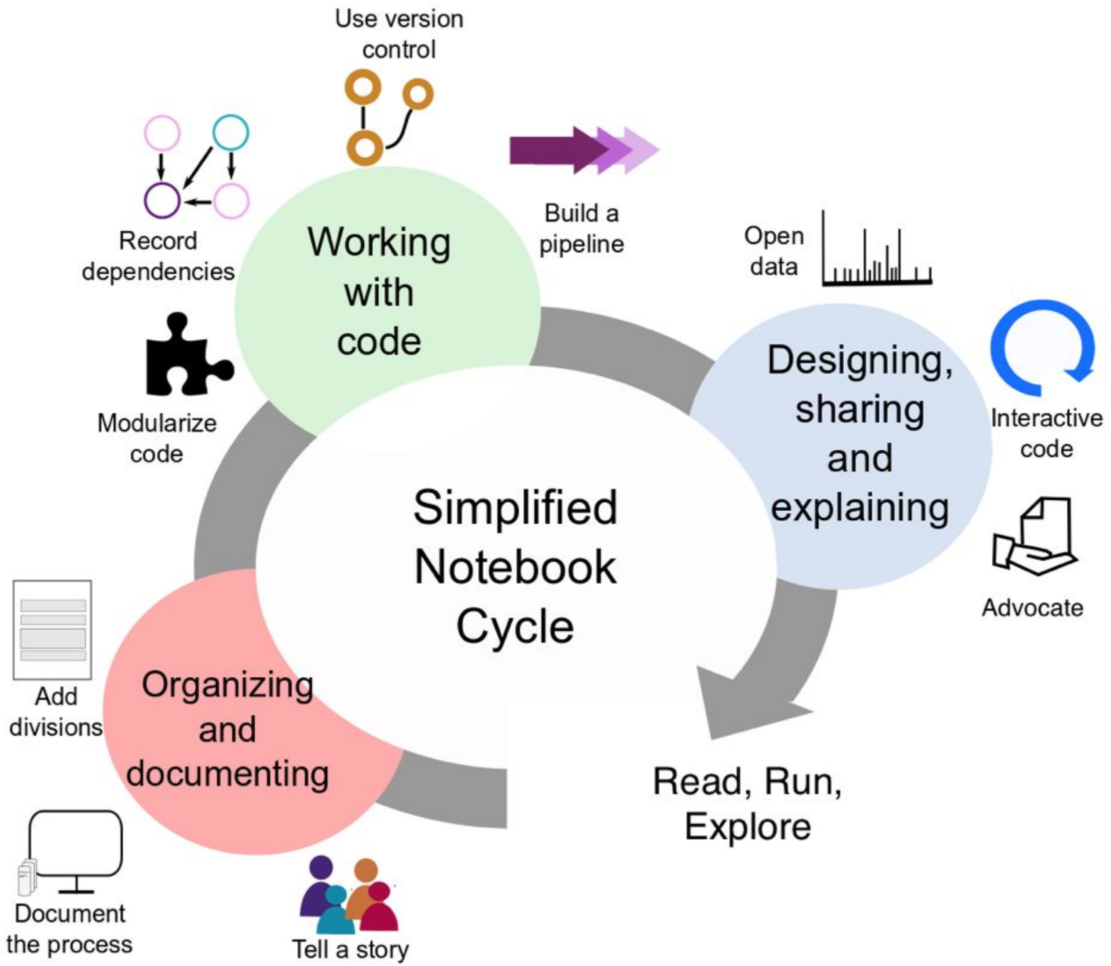
Published: July 25, 2019 • https://doi.org/10.1371/journal.pcbi.1007007

229 Save	34 Citation
43,128 View	1,051 Share

Paper: https://doi.org/10.1371/journal.pcbi.1007007

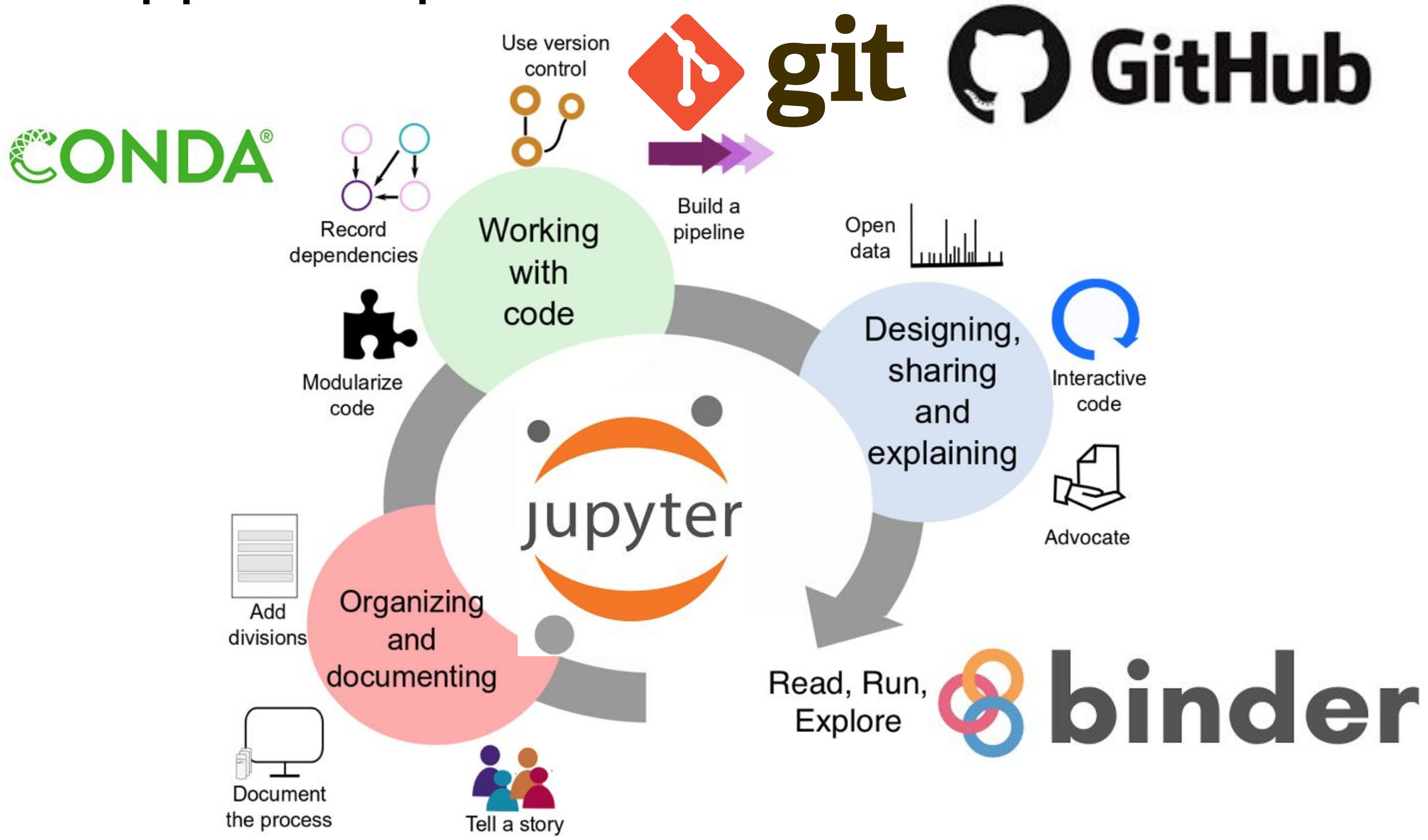
Git repo: https://github.com/jupyter-guide/ten-rules-jupyter

Ten Simple Rules

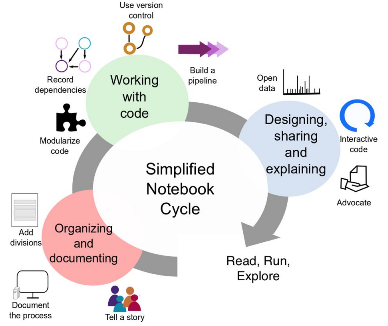


Ten Simple Rules for Writing and Sharing Computational Analyses in Jupyter Notebooks, PLOS Comp. Biol. 2019, https://doi.org/10.1371/journal.pcbi.1007007

Tools to Support Reproducible Workflows

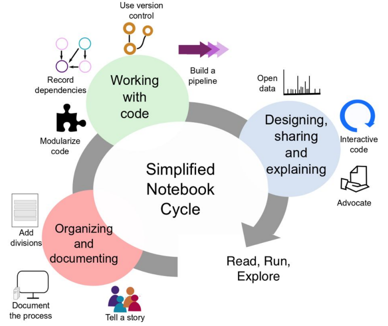


Organizing and Documenting



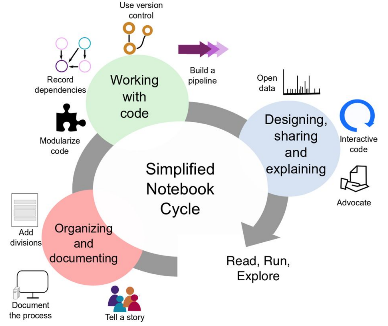
- Rule 1: Tell a Story for an Audience
  - Beginning - introduce topic
  - Middle - describe steps
  - End - interprets results
  - Describe not just what you did, by why you did it, how the steps are connected, and what it all means.
  - Adjust your description depending on the intended audience
- Rule 2: Document the process, not just the results
  - Add descriptive notes, e.g., why a particular parameter was chosen
- Rule 3: Use cell divisions to make steps clear
  - Avoid long cells
  - Limit each cell to one meaningful step
  - Split long notebooks into a series of notebooks
  - Keep a top-level index notebook with links to the individual notebooks

Working with Code



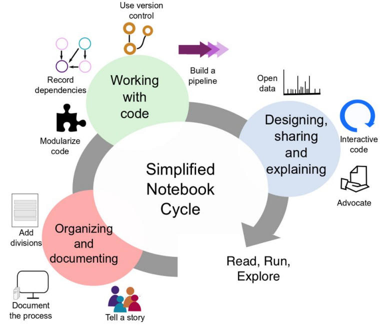
- Rule 4: Modularize Code
  - Use functions instead of duplicating code cells
- Rule 5: Record Dependencies
  - Manage your dependencies explicitly from the start using a tool such as
    - Conda’s environment.yml
    - pip’s requirements.txt
- Rule 6: Use Version Control
  - Consider using a public repository from the beginning of a project
  - Tie research results to specific software versions
- Rule 7: Build a Pipeline
  - Design notebooks with reuse in mind (different input data and parameters)
  - Define key input data and parameters at the top of each notebook
  - Break long notebooks into smaller notebooks that focus on one or a few analysis steps.

Sharing, explaining



- Rule 8: Share and Explain Your Data
    - Share your data in a repository with a persistent identifier, e.g., DOI or ARK
        - Bio repositories, e.g., NCBI, Ensemble, PDB
        - General repositories, e.g., Zenodo https://zenodo.org/
    - Small datasets can be stored in GitHub with your source code (< 50MB)
        - E.g., in a /data folder
    - Very large datasets
        - Consider using a sample of the data and a link to the original data
    - Save intermediate data after data processing
        - E.g., in /intermediate_data folder
        - Can be used to verify each step in a workflow

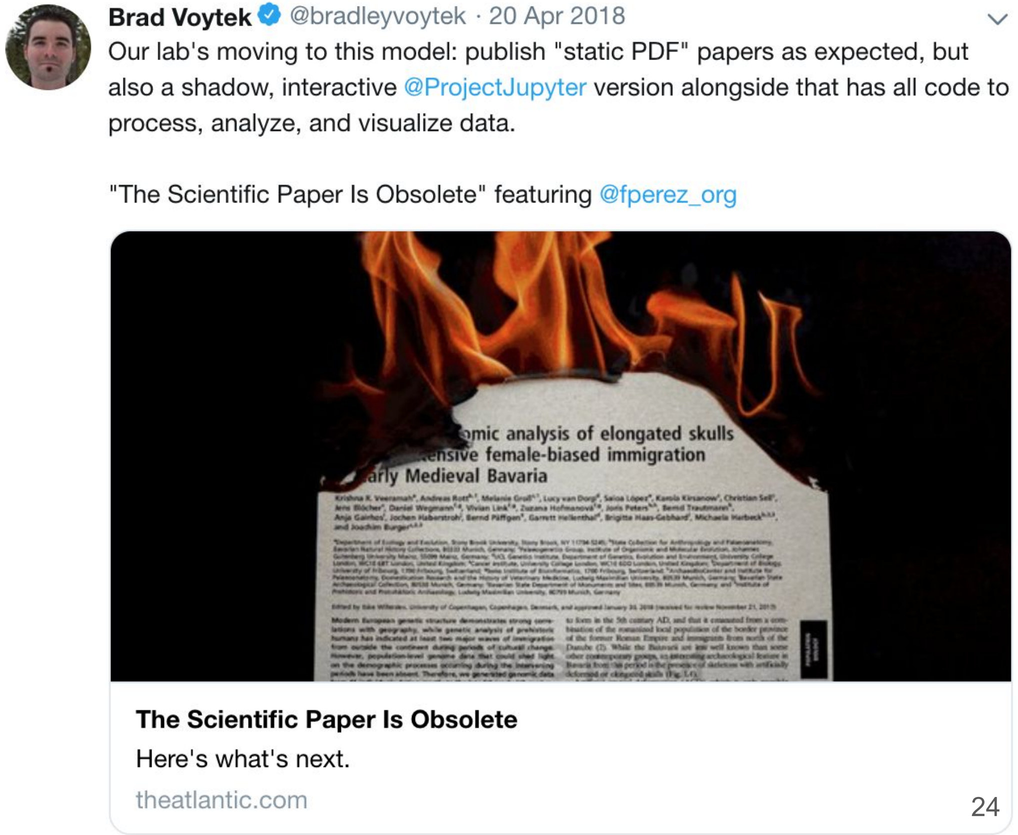
Sharing, explaining cont.



- Rule 9: Design your notebooks to be read, run, and explored
  - Git repository
    - Add a descriptive README file
    - Add a LICENCE file (liberal licence, e.g., MIT, Apache 2)
    - Add a static HTML/PDF file of your notebooks for long-term preservation
    - Add Binder badge/link to launch notebooks in the cloud (https://mybinder.org/)
  - Consider using ipywidgets to add menus or sliders to enable interactive exploration of parameters

Sharing, explaining cont.

- Rule 10: Advocate for open research
  - Apply what you learned in this tutorial in your own research and be an advocate for open and reproducible research in your lab or workplace
  - Publish a fully reproducible paper! Create all figures, data tables, and all other computational results using Jupyter Notebook and deposit in Github.


omic analysis of elongated skulls
ensive female-biased immigration
arly Medieval Bavaria

Brad Voytek @bradleyvoytek · 20 Apr 2018

Our lab's moving to this model: publish "static PDF" papers as expected, but also a shadow, interactive @ProjectJupyter version alongside that has all code to process, analyze, and visualize data.

"The Scientific Paper Is Obsolete" featuring @fperez_org

The Scientific Paper Is Obsolete

Here's what's next.

theatlantic.com

Enabling Reproducibility and Reusability

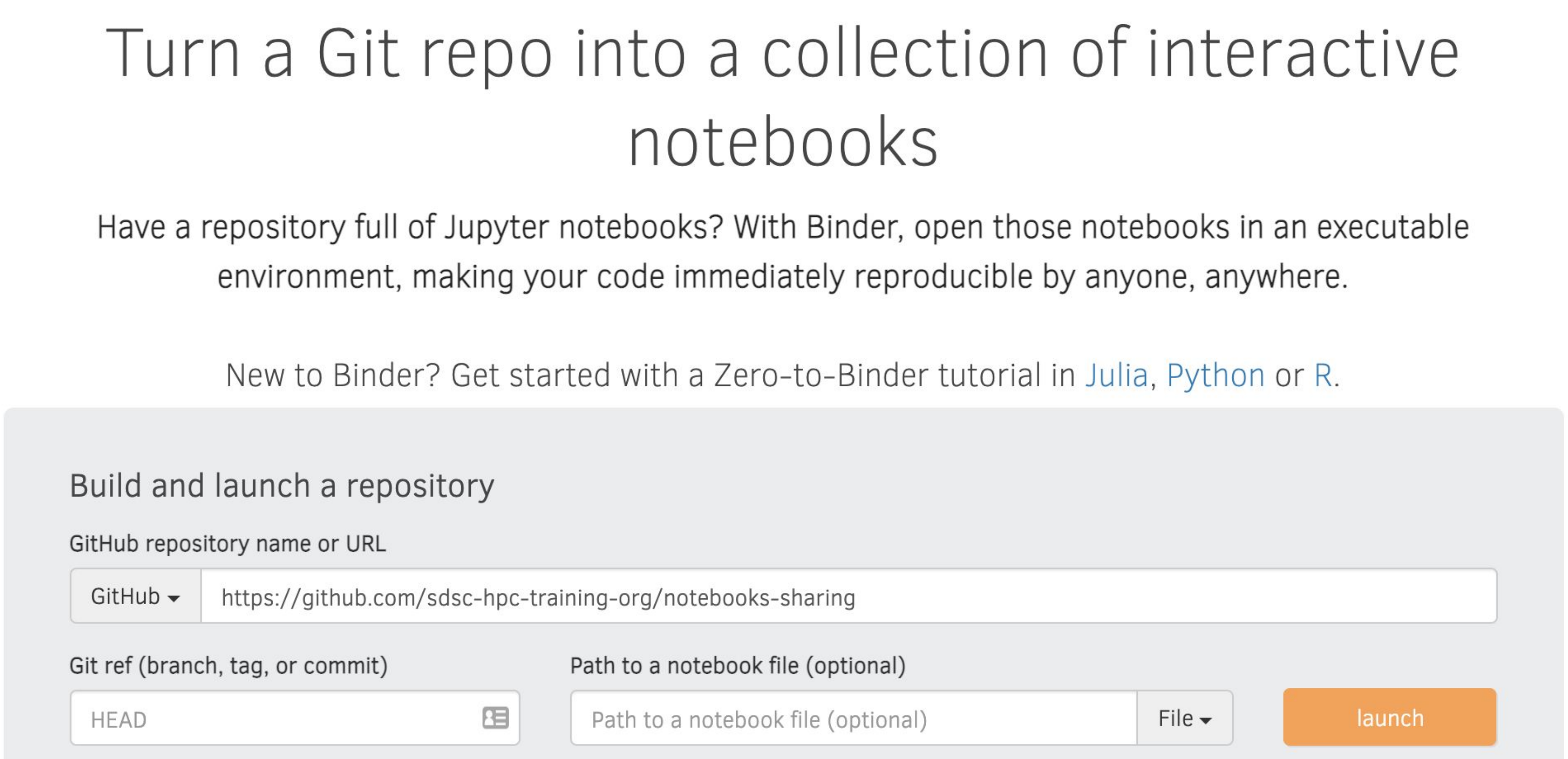
Hosting runnable Notebooks in the Public Cloud (for free)

The binder Project

A community that builds free and open-source tools for reproducible, sharable scientific environments that are workflow- and platform-agnostic.

Source: Chris Holdgraf, bit.ly/2019-elife-cc-holdgraf binder

https://mybinder.org/



Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

New to Binder? Get started with a Zero-to-Binder tutorial in Julia, Python or R.

Build and launch a repository

GitHub repository name or URL

GitHub ▾ | https://github.com/sdsc-hpc-training-org/notebooks-sharing

Git ref (branch, tag, or commit): HEAD

Path to a notebook file (optional): Path to a notebook file (optional) | File ▾

Public Cloud Environments

Platform	URL	Memory	Cores	Use for	Comments	Account
MyBinder	https://mybinder.org/	2GB	1	small examples	some ports are blocked	no
Pangeo Binder	https://binder.pangeo.io/	32GB (?)	6 (?)	when exceeding MyBinder limits	open ports, e.g., FTP	no
Google Colab	https://research.google.com/colaboratory/	variable	?	GPU/TPU	software installations using pip in Notebook, share notebooks on Google Drive	yes
CyVerse	https://cyverse.org/discovery-environment	per request	per request	store notebooks, results, and data	100GB storage	yes

Set up for Hands-on Session On Thur. June 24

- Sign up for a free personal GitHub account
    - https://docs.github.com/en/get-started/signing-up-for-github/signing-up-for-a-new-github-account
- Run the software installation and run example to test installation
    - https://github.com/ciml-org/ciml-summer-institute-2021/tree/main/1.4_installing_your_own_miniconda_creating_conda_environments#14_installing_your_own_miniconda_creating_conda_environments
- If you run into problems, describe your problem or put screen shot in the “conda-notebooks” Slack channel and mention your Expanse account name (xdtrXX)

- If you want run of you own projects, follow the same workflow (e.g., create an environment.yml file for you project)

Questions?