Preparing code and data for reproducible publication: a hands-on tutorial

JCDL 2020
August 3, 2020

William A. Ingram and Edward A. Fox
Virginia Tech
Blacksburg, VA 24061
USA

https://bit.ly/jcdl-reproducibility-pptx

Instructors

- **William A. (Bill) Ingram**
  - M.S., L.I.S. from iSchool at UIUC; Ph.D. (in progress) Computer Science at Virginia Tech
  - Assistant Dean, Virginia Tech University Libraries
    - Overseeing: Digital Libraries, IT Services, Digital Preservation and Imaging, Special Collections and University Archives, Policy and Governance, Finance
  - ~20 years IT experience
  - ~15 years in libraries, mostly in digital libraries

Instructors

- **Dr. Edward A. Fox**
  - Ph.D. and M.S. in Computer Science from Cornell University, and a B.S. from M.I.T.
  - Fellow of both ACM and IEEE, cited for leadership in digital libraries and information retrieval
  - Executive Director and Chairman of the Board of the Networked Digital Library of Theses and Dissertations
  - Professor of CS at Virginia Tech
  - Former Chair (now a member) of the JCDL Steering Committee as well as of the IEEE-CS Technical Committee on Digital Libraries

Tutorial History

A version of this tutorial was given at JCDL 2019, June 2–6, 2019, in Urbana-Champaign, Illinois, USA, by April Clyburne-Sherin and Xu Fei.

Disclaimer: April Clyburne-Sherin and Xu Fei were employees of Code Ocean, a software service featured prominently in this tutorial.

After adapting her materials and obtaining her guidance, Ingram and Fox gave a version of this tutorial at ETD2019, November 6–8, 2019, in Porto, Portugal.

Thanks go to April Clyburne-Sherin and Xu Fei for their prior assistance.

Workshop POP

- **Purpose:** To introduce skills and tools in organization, documentation, automation, containerization, and dissemination of research.

- **Outcome:** You feel more confident applying relevant skills and tools to guide the sharing of your research code and data.

- **Process:** You adapt & apply some skills or tools we discuss today next time your share or publish your research.

Agenda

Introduction

Organization
- Exercise 1: One repository
- Exercise 2: Separate code & data

Documentation
- Exercise 3: Document data & code
- Exercise 4: Specify run environment
- Exercise 5: Specify dependencies

Automation
- Exercise 6: Containerization
- Exercise 7: Create a master script
- Exercise 8: Create relative paths

Dissemination
- Exercise 9: Specify a license
- Exercise 10: Share your code!

Schedule

Beijing Time (UTC+8)

- 0:30-2:00 — Part 1
- 2:00-2:30 — Coffee/Tea Break
- 2:30-4:00 — Part 2

We will take a short break midway through each part, but free to take additional breaks you need.

Your thoughts?

Is there a reproducibility crisis?

A crisis? *(Nature 2016)*

Monya Baker. 2016. 1,500 scientists lift the lid on reproducibility. Nature News 533, 7604 (May 2016), 452. DOI:https://doi.org/10.1038/533452a

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

An opportunity to help your future self

"It takes some effort to organize your research to be reproducible… the principal beneficiary is generally the author herself."- Schwab & Claerbout

See: Making Scientific Contributions Reproducible

Your experience

Have you failed to reproduce an experiment?

- How many participants have had difficulty reproducing someone else's work?
- How many participants have had difficulty reproducing your own work a few weeks, months, or years later?

Defining reproducibility

ACM Definitions: https://www.acm.org/publications/policies/artifact-review-badging

- *Repeatability:* Same team; same experimental setup
- *Reproducibility*: Different team; same experimental setup
- *Replicability*: Different team; different experimental setup

Our definition of *computational reproducibility* — based Victoria Stodden et al. (2014):

The calculation of quantitative scientific results by an independent person or group using the original datasets and methods

Computational reproducibility **depends** on open code and data.

Computational reproducibility

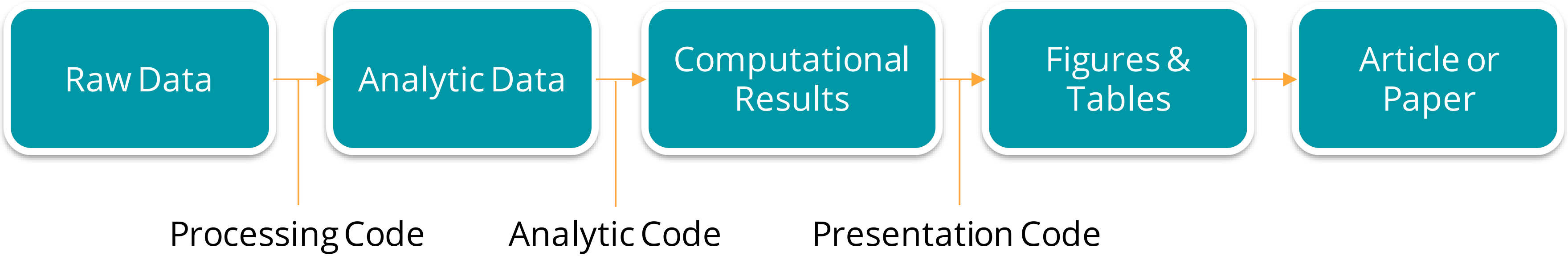
"An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the **complete software development environment and the complete set of instructions** which generated the figure."

- Buckheit and Donoho (1995)'s distillation of Claerbout and Karrenbach (1992)

Jon F Claerbout and Martin Karrenbach. 1992. Electronic documents give reproducible research a new meaning. In SEG technical program expanded abstracts 1992. Society of Exploration Geophysicists, 601–604.

Jonathan B Buckheit and David L Donoho. 1995. Wavelab and reproducible research. In Wavelets and statistics. Springer, 55–81.

Research data pipeline



Roger Peng. 2015. Report writing for data science in R. Leanpub. Retrieved from https://leanpub.com/reportwriting

Provenance tracking is important document the full chain of computational events along the research pipeline.

https://bit.ly/jcdl-reproducibility-pptx

Technical barriers to computational reproducibility

- "Dependency Hell"
  - Frustration with software packages which have dependencies on specific versions of other software packages
- Imprecise documentation
  - Or none at all
- Code rot
  - Dependency packages no longer available
- Barriers to adoption
  - New technologies to learn and associated opportunity costs

Carl Boettiger. 2015. An introduction to Docker for reproducible research. SIGOPS Oper. Syst. Rev. 49, 1 (January 2015), 71–79. doi:10.1145/2723872.2723882, arXiv:1410.0846

Reproducibility is a spectrum

Roger D. Peng. 2011. Reproducible Research in Computational Science. Science 334, 6060 (December 2011), 1226–1227. DOI:https://doi.org/10.1126/science.1213847

See: Fig. 1. The spectrum of reproducibility

The Practice of Reproducible Research

Justin Kitzes, Daniel Turek, and Fatma Deniz (Eds.). 2017. The practice of reproducible research: case studies and lessons from the data-intensive sciences. Univ of California Press. Retrieved from https://www.practicereproducibleresearch.org/

"At a beginning level, the first of these practices largely involves placing files in a **clear directory structure** and **creating metadata** to describe them. The second is met by writing code, or scripts, to **perform each step automatically**, or where this is not possible, **documenting all manual steps** needed to complete a task at a level that would allow a second researcher to unambiguously repeat them. The third is met through the **overall workflow design**, especially a clear conceptualization of the different operations that need to occur sequentially and how they support each other. ... Crucial to reproducing a study is **providing sufficient details** about its execution through **reports, papers, lab notebooks, etc.**"

Questions?

https://bit.ly/jcdl-reproducibility-pptx

Agenda

Introduction

Organization
- Exercise 1: One repository
- Exercise 2: Separate code & data

Documentation
- Exercise 3: Document data & code
- Exercise 4: Specify run environment
- Exercise 5: Specify dependencies

Automation
- Exercise 6: Containerization
- Exercise 7: Create a master script
- Exercise 8: Create relative paths

Dissemination
- Exercise 9: Specify a license
- Exercise 10: Share your code!

We can **organize for reproducibility**:

- **Archive the exact versions** of data used and include them in your repository.
- **Bundle dependencies** and include them in your repository rather than retrieve on demand.
- **Link to repositories**.

Woodbridge et al. analyzed the validity of a small sample of Jupyter notebooks associated with papers in PubMed Central.

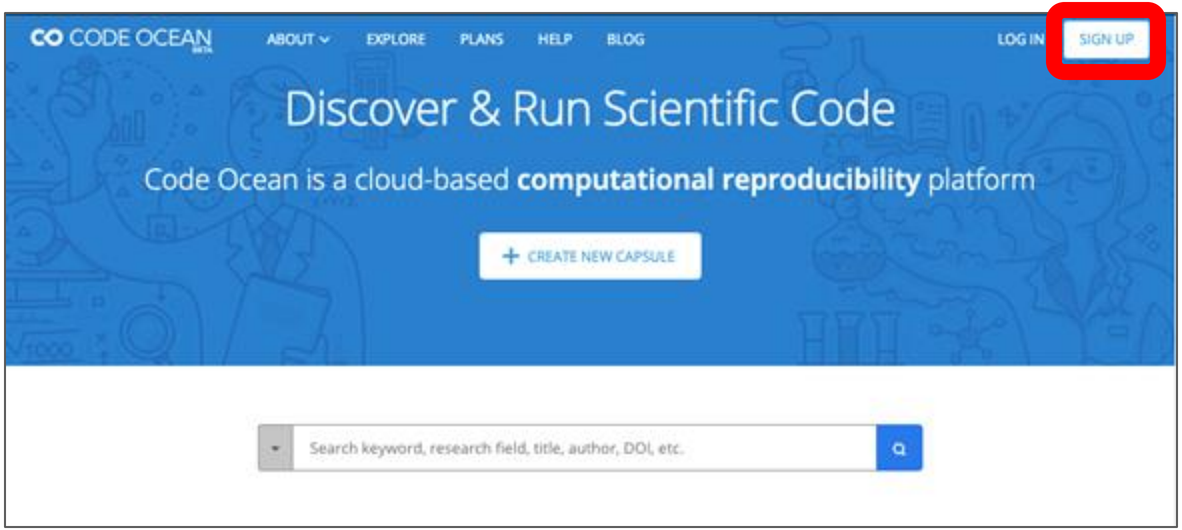
- Files, data, dependencies needed to execute analyses **were often missing**.
- They were able to successfully execute **only one** of the ~25 notebooks that they tested.

Mark Woodbridge. 2017. Jupyter Notebooks and reproducible data science, https://markwoodbridge.com/2017/03/05/jupyter-reproducible-science.html.

Exercise 1:

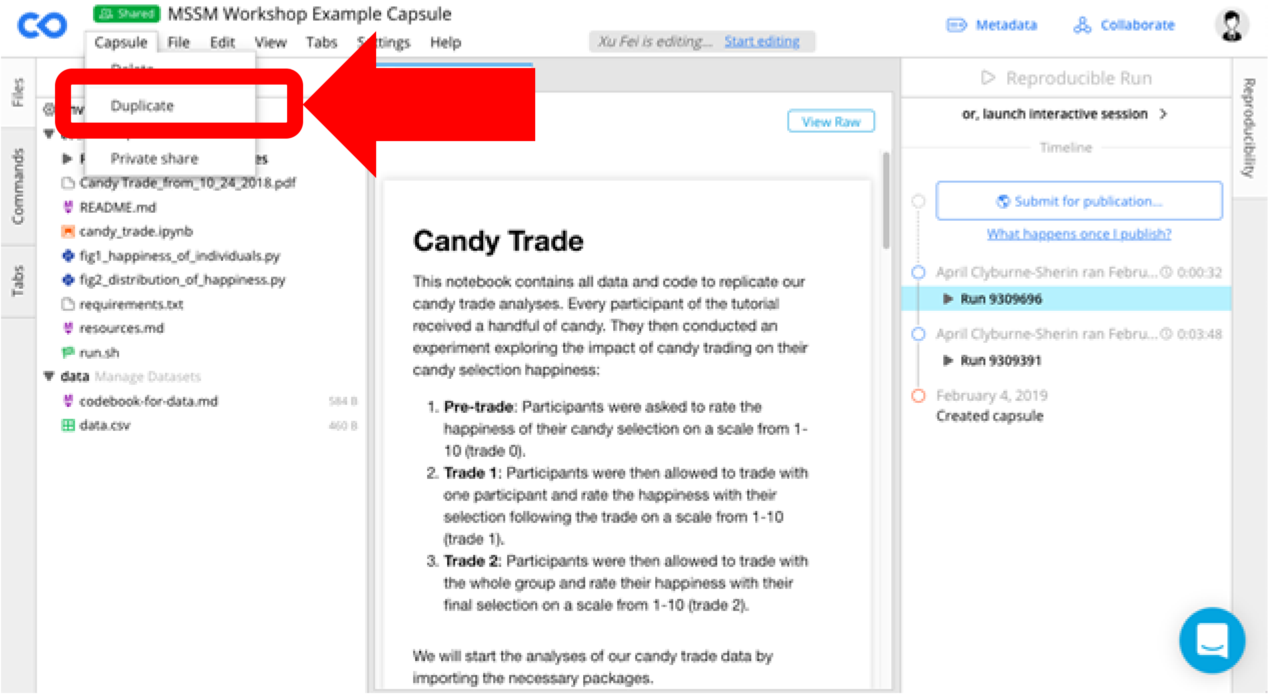
- **Create one repository that holds all related research files:**
  - Data
  - Code
  - Notebooks
  - Documentation
  - etc.

Create a Code Ocean account



- https://codeocean.com/
- You can **delete it and opt out of any communications** if you wish! For completing the exercises only. :)
- You will need to verify your email address

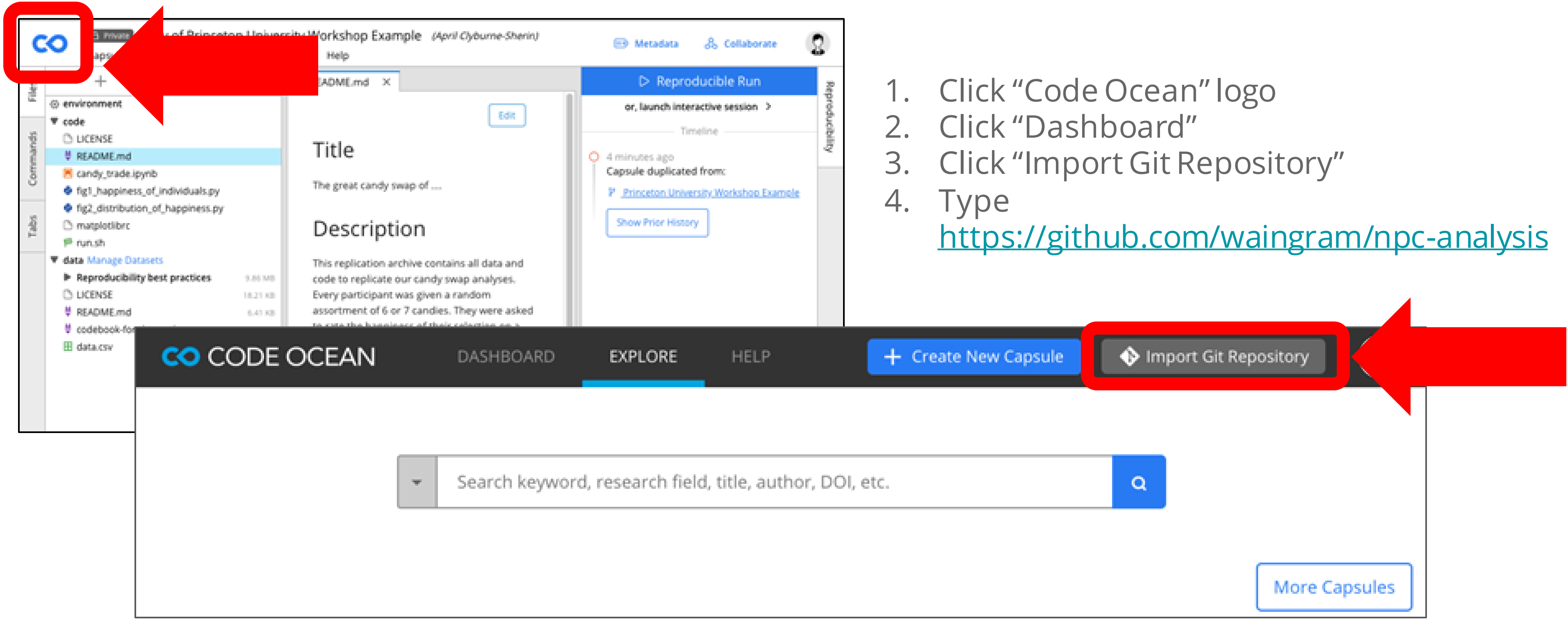
Duplicate this capsule: http://bit.ly/etd-example



- Click “Capsule”
- Select “Duplicate”

Create a new compute capsule

https://github.com/waingram/npc-analysis



1. Click "Code Ocean" logo
2. Click "Dashboard"
3. Click "Import Git Repository"
4. Type https://github.com/waingram/npc-analysis

https://bit.ly/jcdl-reproducibility-pptx

Organization

"Perhaps the most important step to take towards ease of reproducibility is to be *organized*."

Broman, K. Initial steps toward reproducible research. http://kbroman.org/steps2rr/ (2016).

Exercise 2:

- **Organize your research to separate code from data.**

```
.
|-- CITATION
|-- README
|-- LICENSE
|-- requirements.txt
|-- data
|     -- birds_count_table.csv
|-- doc
|     -- notebook.md
|     -- manuscript.md
|     -- changelog.txt
|-- results
|     -- summarized_results.csv
|-- src
|     -- sightings_analysis.py
|     -- runall.py
```

Resource on reproducible organization:

- Karl Broman: http://kbroman.org/steps2rr/pages/organize.html

Reference Slide

Checklist

```
/ [root]
├── code
│   ├── my_algorithm.py
│   ├── README.md
│   ├── run.sh
│   └── ...
├── data
│   ├── my_data.csv
│   ├── my_sample_image.png
│   └── ...
└── results
    └── [your future results]
```

- Create one repository or directory that holds all related research files.
- Organize your research to separate data, code, and results.
- Save results explicitly.
- Identify a strategy for sensitive data.

Tools

OSF

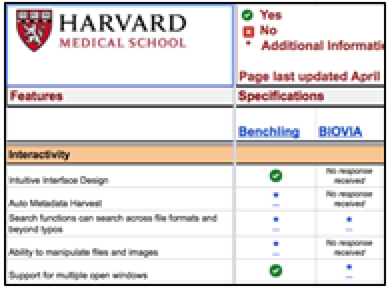
GitHub

CODE OCEAN BETA

SCINOTE

- Open Science Framework: collaborative project organization tool
- GitHub: collaborative coding, and project management
- eLNs: free or paid, lab organization
- Code Ocean: built in best practices

Resources



- Strategies for sensitive data sharing: [Code Ocean Summary]
- Harvard eLN Features Matrix: https://docs.google.com/spreadsheets/d/1ar8fgwagOh30E31EAPL-Gorwn_g6XNf81g3VDQnQ_l8/edit?usp=sharing

https://bit.ly/jcdl-reproducibility-pptx

Document your data

- Make a codebook or data dictionary.
    - Document each element or variable
    - in your dataset and data model.
- Resources on making a great codebook or data dictionary:
    - DataONE: https://www.dataone.org/best-practices/create-data-dictionary
    - McGill Codebook Cookbook: http://www.medicine.mcgill.ca/epidemiology/joseph/pbelisle/CodebookCookbook.html
    - UPenn: https://guides.library.upenn.edu/datamgmt/documentation
    - Karl Broman: http://kbroman.org/dataorg/pages/dictionary.html
- Example codebook:
    - AJPS Replication Package: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EZSJ1S

Create a README

- Create a README.txt or README.md.
- Resource on making a great README file:
  - Cornell (includes a template): https://data.research.cornell.edu/content/readme
- Resource on using markdown:
  - GitHub: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Document your code: Literate programming

- Term coined by Donald Knuth in 1992
- Interleave narrative text and computer code in the same document
- KnitR, RMarkdown, Sweave
- Jupyter Notebooks

Demo:

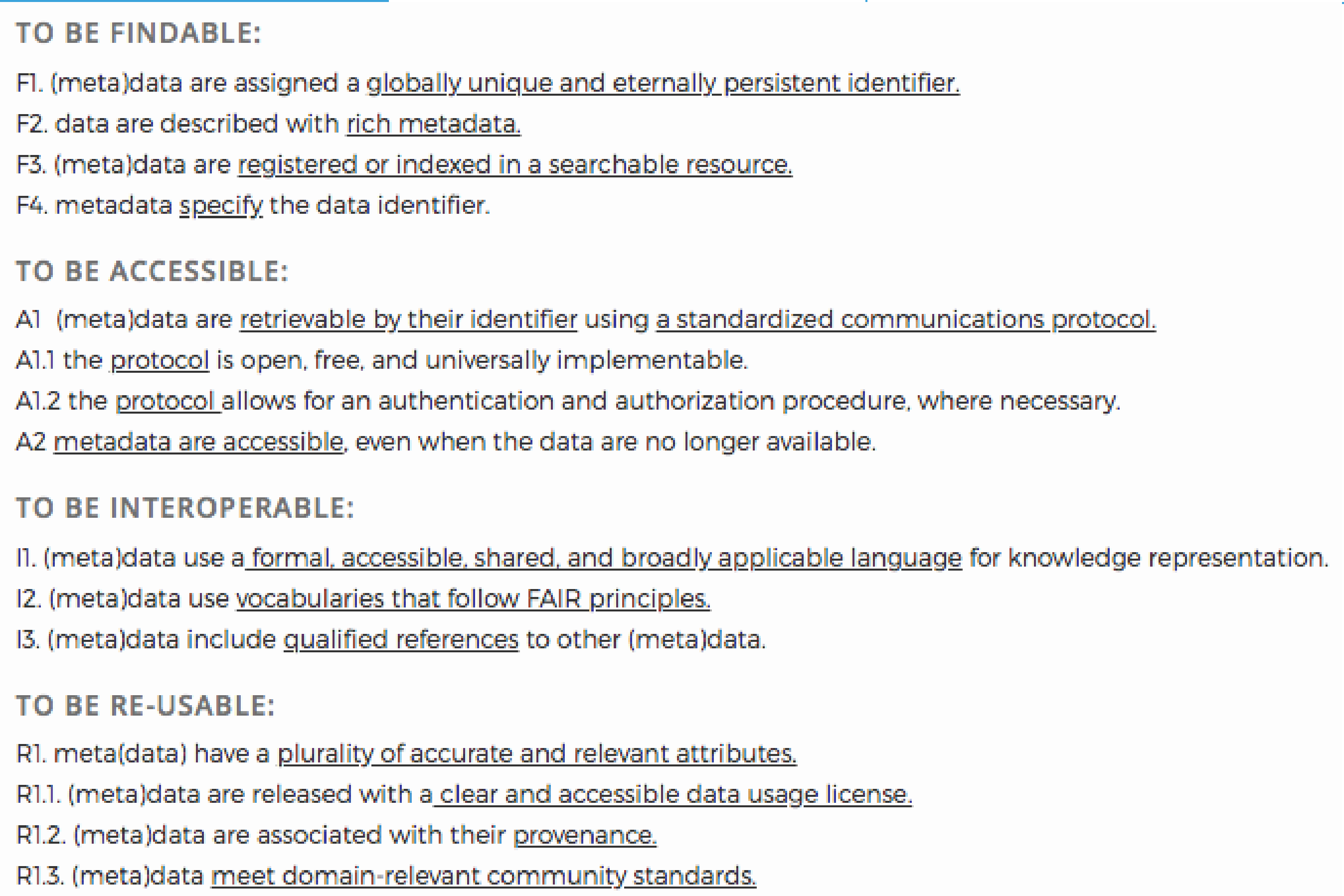
- **Consider using literate programming to document the analysis narrative with the code.**
  - Jupyter Notebook
  - RMarkdown

Explore Jupyter notebooks in this example capsule: http://bit.ly/jcdl-example

Explore RMarkdown in this example capsule: http://bit.ly/rmarkdown-example

Follow the FAIR Principles:

Findable
Accessible
Interoperable
Reusable



TO BE FINDABLE:

F1. (meta)data are assigned a globally unique and eternally persistent identifier.
F2. data are described with rich metadata.
F3. (meta)data are registered or indexed in a searchable resource.
F4. metadata specify the data identifier.

TO BE ACCESSIBLE:

A1 (meta)data are retrievable by their identifier using a standardized communications protocol.
A1.1 the protocol is open, free, and universally implementable.
A1.2 the protocol allows for an authentication and authorization procedure, where necessary.
A2 metadata are accessible, even when the data are no longer available.

TO BE INTEROPERABLE:

I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
I2. (meta)data use vocabularies that follow FAIR principles.
I3. (meta)data include qualified references to other (meta)data.

TO BE RE-USABLE:

R1. meta(data) have a plurality of accurate and relevant attributes.
R1.1. (meta)data are released with a clear and accessible data usage license.
R1.2. (meta)data are associated with their provenance.
R1.3. (meta)data meet domain-relevant community standards.

https://bit.ly/jcdl-reproducibility-pptx

- https://www.go-fair.org/fair-principles/

Exercise 3:

- **Create a README file and data dictionary.**

Documenting your file overview and dependencies in your README:

- AJPS Replication Package: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EZSJ1S

Documenting your data in a codebook or data dictionary:

- DataONE: https://www.dataone.org/best-practices/create-data-dictionary

Resource on using markdown:

- GitHub: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Lessons learned: testing computational reproducibility

PMC “jupyter OR ipynb” -> 107 papers

“My initial thought was that analysing the validity of the notebooks would simply involve searching the text of each article for a notebook reference, then downloading and executing it … It turned out that this was hopelessly naive…”

Jupyter Notebooks and reproducible data science

Introduction

One of the ideas pitched by Daniel Mietchen at the London Open Research Data do-a-thon for Open Data Day 2017 was to analyse Jupyter Notebooks mentioned in PubMed Central. This is potentially valuable exercise because these notebooks are an increasingly popular tool for documenting data science workflows used in research, and therefore play an important role in making the relevant analyses replicable.

Mark Woodbridge, Daniel Sanz, Daniel Mietchen, & Ross Mounce (2017). Jupyter Notebooks and reproducible data science, https://markwoodbridge.com/2017/03/05/jupyter-reproducible-science.html.

Specify your environment and package versions

- Specify your environment and package versions.
  - Example in R: use sessionInfo() to specify your environment and package versions.
  - Example in Python: use pip freeze > requirements.txt
- Add these to your README or create a requirements.txt file.
- Example of documenting packages in your README:
  - AJPS Replication Package: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/EZSJ1S
- Example of documenting dependencies:
  - Binder: http://mybinder.readthedocs.io/en/latest/using.html#preparing-a-repository-for-binder

Dependencies in Python

```
###### Requirements without Version Specifiers ######`
nose
nose-cov
beautifulsoup4

###### Requirements with Version Specifiers ######`
docopt == 0.6.1             # Version Matching. Must be version 0.6.1
keyring >= 4.1.1            # Minimum version 4.1.1
coverage != 3.5             # Version Exclusion. Anything except version 3.5
Mopidy-Dirble ~= 1.1        # Compatible release. Same as >= 1.1, == 1.*
```

Example Requirements File — pip 9.0.1 documentation

Exercise 4:

- **Specify the run environment for your analyses.**

Example: **Base Environment: Python 3.7.0**

Exercise 5:

- **Specify your packages and dependencies with versions.**
  - pip freeze > ../results/requirements.txt

Resource on documenting dependencies:

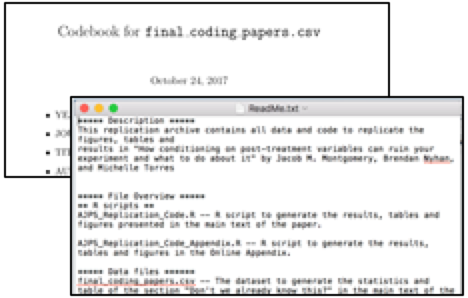
- Binder: http://mybinder.readthedocs.io/en/latest/using.html#preparing-a-repository-for-binder

Example: **conda installer: jupyter 1.0.0, numpy, pandas, matplotlib**

Organization | **Documentation** | Automation | Dissemina

Reference Slide

Checklist



- Consider literate programming.
- Document each element or variable in your dataset with a data dictionary / codebook.
- Create a README file.

Tools

GitHub

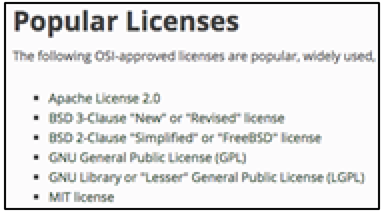
jupyter

CODE OCEAN BETA

- Version control: git and GitHub tracks changes to documents and metadata
- Literate programming: knits documentation with code (Jupyter)
- Document & share metadata: Code Ocean renders documentation, notebooks, and records metadata

Resources

DCC because good research needs good data



- DataONE: https://www.dataone.org/best-practices/create-data-dictionary
- Cornell: https://data.research.cornell.edu/content/readme
- Digital Curation Center: http://www.dcc.ac.uk/resources/how-guides/license-research-data
- OSI: https://opensource.org/licenses

https://bit.ly/jcdl-reproducibility-pptx

Questions?

https://bit.ly/jcdl-reproducibility-pptx

BREAK

https://bit.ly/jcdl-reproducibility-pptx

Agenda

Introduction

Organization

- Exercise 1: One repository
- Exercise 2: Separate code & data

Documentation

- Exercise 3: Document data & code
- Exercise 4: Specify run environment
- Exercise 5: Specify dependencies

Automation

- Exercise 6: Containerization
- Exercise 7: Create a master script
- Exercise 8: Create relative paths

Dissemination

- Exercise 9: Specify a license
- Exercise 10: Share your code!

What *Woodbridge et al.* found:

- **Manual manipulation or setup** was needed to reproduce results, often without documentation of how the results were produced.

DevOps

- Short for **Dev**elopment and Systems **Op**eration
- Practice depends on *scripting*, rather than *documenting*, to set up the development environment of the original researchers
- E.g., Makefiles, bash scripts
- Adds complexity
- Researchers might lack the necessary technical skills

Clark, Dav et al. "BCE: Berkeley's Common Scientific Compute Environment for Research and Education". In Proceedings of the 13th Python in Science Conference, 2014. https://research-it.berkeley.edu/publications/bce-berkeley%E2%80%99s-common-scientific-compute-environment-research-and-education

Docker

- https://docs.docker.com/
- Docker containers allow researchers to share prebuilt application runtime environments (including dependencies) from one machine to another.
- Lighter and faster than VMs, containers do not include a full operating system.

Carl Boettiger. 2015. An introduction to Docker for reproducible research. SIGOPS Oper. Syst. Rev. 49, 1 (January 2015), 71–79. DOI:https://doi.org/10.1145/2723872.2723882, https://arxiv.org/abs/1410.0846

Olivier Mesnard and Lorena A. Barba. 2020. Reproducible Workflow on a Public Cloud for Computational Fluid Dynamics. Computing in Science Engineering 22, 1 (January 2020), 102–116. DOI:https://doi.org/10.1109/MCSE.2019.2941702, http://arxiv.org/abs/1904.07981

The terms:

- **Dockerfile**: Readable instructions for how to build an image.
- **Image**: Everything your application needs to run, all bundled together (includes Dockerfile, libraries, and code).
- **Layer**: A Dockerfile directs Docker to build the initial image layer from a base image, and then other layers are built on top.
- **Container**: Started and created from an image.
- **Registry**: Images are stored and retrieved from registries.

Hale, Jeff. *Learn Enough Docker to be Useful*. https://towardsdatascience.com/learn-enough-docker-to-be-useful-b7ba70caeb4b

The metaphor: PIZZA!

- **Dockerfile**: The recipe.
- **Image**: The recipe and the ingredients combined as an all-in-one pizza-making-kit.
- **Layer**: The ingredients are the layers. You've got crust, sauce, and cheese for this pizza.
- **Container**: Cooked pizza. Cooked by Docker (the oven).
- **Registry**: All-in-one pizza-making-kit factories?

Hale, Jeff. *Learn Enough Docker to be Useful*. https://towardsdatascience.com/learn-enough-docker-to-be-useful-b7ba70caeb4b

Containers solve:

- Dependency Hell - install, error, google, install, error…
  - Provides other researchers with a binary image in which all the software has already been installed, configured, and tested.
- Imprecise documentation - missing installation info.
  - Dockerfile provides a human readable summary of the necessary software dependencies needed to execute the code. Dependencies are automatically documented as they are installed.
- Code rot - dependencies change, the code breaks
  - Reduced risk with by archiving images

Boettiger, Carl. *An introduction to Docker for reproducible research.* [10.1145/2723872.2723882](https://doi.org/10.1145/2723872.2723882)

We can **publish using containers**:

- Use container technology to **directly express dependencies.**
- **Configure an image** for your analyses with Docker, binder, WholeTale, or Code Ocean.

Online Container Platforms

- Services include: Binder, Code Ocean, Colaboratory, Gigantum and Nextjournal
- Allow researchers to run code in the cloud
- Easier to use than installing Docker locally
- Several journals now use Code Ocean for peer review and to promote computational reproducibility

Perkel, J. M. (2019). Make code accessible with these cloud services. Nature, 575(7781), 247. doi:10.1038/d41586-019-03366-x

Exercise 6:

- **Use container technology to create an image of your complete computational environment.**
  - Code Ocean
  - Binder

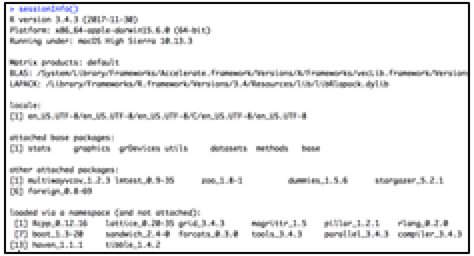
Export your capsule to see how an image and Dockerfile were created through your specifications.

Inspect the Dockerfile. Inspect the requirements.txt file.

Build a container with repo2docker using mybinder and your requirements.txt file.

Reference Slide

Checklist



- Specify your computational environment and package versions.
- Configure a container to make your analysis portable and reusable.

Tools

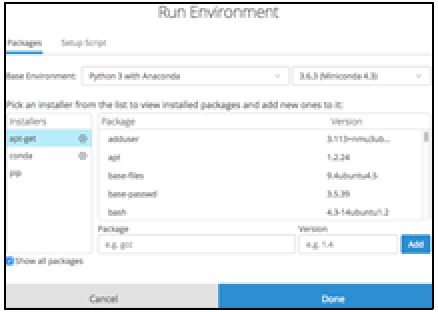
CODE OCEAN

docker

binder (beta)

- Container technology: packages data, code, metadata, & computational environment for portable analyses
- Docker: container technology for devs
- Code Ocean: easy configuring, preservation, & reuse of containers for researchers
- Binder: configure & share containers

Resources



- Documenting dependencies: http://mybinder.readthedocs.io/en/latest/using.html#preparing-a-repository-for-binder
- Specifying environments: https://help.codeocean.com/getting-started/the-computational-environment/selecting-a-base-environment

https://bit.ly/jcdl-reproducibility-pptx

Questions?

https://bit.ly/jcdl-reproducibility-pptx

Agenda

Introduction

Organization

- Exercise 1: One repository
- Exercise 2: Separate code & data

Documentation

- Exercise 3: Document data & code
- Exercise 4: Specify run environment
- Exercise 5: Specify dependencies

Automation

- Exercise 6: Containerization
- Exercise 7: Create a master script
- Exercise 8: Create relative paths

Dissemination

- Exercise 9: Specify a license
- Exercise 10: Share your code!

`https://bit.ly/jcdl-reproducibility-pptx`

We can **automate the execution of our analyses**:

- Create a master script to execute all analyses.
- Reproduce results automatically as a function of the data & the code; Save results explicitly.
- Use relative paths.

Exercise 6:

- **Create a master script to execute your code.**

- Explore the file "run.sh".
- Use nbconvert to render your notebook.
  - In your run.sh script, use nbconvert to execute your notebook into the results directory.

- Case study:
  https://bids.gitbooks.io/the-practice-of-reproducible-research/core-chapters/3-basic.html

Exercise 7:

- **Change absolute paths to relative paths.**

Resource explaining paths:

- Karl Broman: http://kbroman.org/steps2rr/pages/organize.html

Code testing and continuous integration

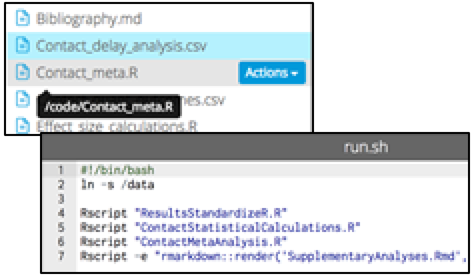
Automate code testing:

- **Automated testing verifies that your software is (relatively) error free.**
    - Most/all software has bugs
    - "ubiquity of error" — Donoho, et al. (2009)
- **Continuous integration systems allow researchers to run code tests frequently and automatically**
    - Travis-CI and Jenkins are popular continuous integration systems

Organization | Documentation | **Automation** | Dissemina

Reference Slide

Checklist



- Use relative rather than absolute paths.
- Create a master script that runs your scripts in sequence.

Tools

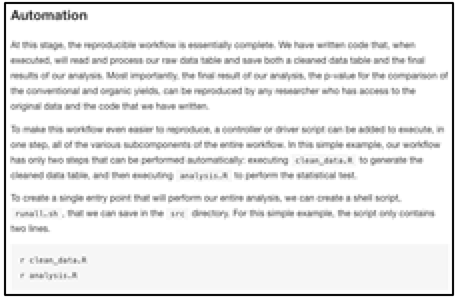
CODE OCEAN

docker

binder (beta)

- Docker: share automated code for devs
- Code Ocean: easy configuring, preservation, & reuse of automated code
- Binder: share automated code for using containers

Resources



Automation

At this stage, the reproducible workflow is essentially complete. We have written code that, when executed, will read and process our raw data table and save both a cleaned data table and the final results of our analysis. Most importantly, the final result of our analysis, the p-value for the comparison of the conventional and organic yields, can be reproduced by any researcher who has access to the original data and the code that we have written.

To make this workflow even easier to reproduce, a controller or driver script can be added to execute, in one step, all of the various subcomponents of the entire workflow. In this simple example, our workflow has only two steps that can be performed automatically: executing `clean_data.R` to generate the cleaned data table, and then executing `analysis.R` to perform the statistical test.

To create a single entry point that will perform our entire analysis, we can create a shell script, `runall.sh`, that we can save in the `src` directory. For this simple example, the script only contains two lines.

```
r clean_data.R
r analysis.R
```

- Karl Broman on paths: http://kbroman.org/steps2rr/pages/organize.html
- Resource on automation using a master script: https://www.practicereproducibleresearch.org/core-chapters/3-basic.html

https://bit.ly/jcdl-reproducibility-pptx

License your work

Add a license.txt file to your project or select one in the metadata section (CO or GitHub)

- Consider Creative Commons licenses for data and text, either CC-0 or CC-BY.
- For software, we recommend a permissive open source license such as the MIT, BSD, or Apache license

Exercise 8:

- **Specify a license for your data and your code.**

Resource on choosing a data licence:

Digital Curation Center: http://www.dcc.ac.uk/resources/how-guides/license-research-data

Resources on choosing a code licence:

- Karl Broman: http://kbroman.org/steps2rr/pages/licenses.html
- Open Source Initiative: https://opensource.org/licenses

What *Woodbridge et al.* found:

- There is no standardized way of **attaching code to published articles**.
- Therefore it is difficult to **discover and retrieve** code.

Persistent Identifiers

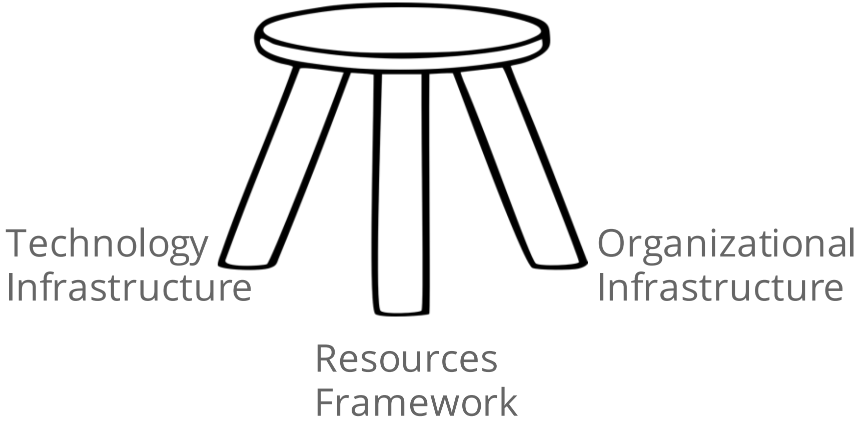
Persistent identifiers increase the reproducibility of the research

- DOI = Digital Object Identifier
- International standard, interoperable, and widely adopted
- DOI contains standard metadata about an object, including the URL to where the object can be found online
- If the URL changes, the metadata can be updated so that the DOI resolves to the object's new location
- The use of DOI increases long-term citation, access, and reuse

See: https://www.doi.org/ and https://www.iso.org/standard/43506.html

Digital Preservation

- **Organizational Infrastructure**: policies, procedures, practices, people
- **Technological Infrastructure**: equipment, software, hardware, secure environment
- **Resources Framework**: ongoing and contingency funding for long-term sustainability



Digital Preservation Workshop: http://www.dpworkshop.org/

Be careful where you store your data

- Use data repositories rather than lab group website or researcher's personal website
- University libraries often maintain secure digital preservation repository services and persistent identifiers for research data
- Some commercial services (Code Ocean, Figshare, Zenodo, etc) also provide preservation services
- Cloud storage systems (Box, Dropbox, Google Drive, etc) are not preservation repositories

We can **embed or link code persistently**:

- **Obtain a DOI for your repository and use this link throughout your article.**
  - Example: Github -> Binder/WholeTale -> Zenodo -> DOI linked in article
  - Example: CodeOcean -> DOI in article
- **Cross link repository with published article in metadata of each.**
- **Embed executable capsule within the article.**
  - Example: https://doi.org/10.1017/bpp.2018.25

Exercise 9:

- **Share your code!**

- Check whether your container is ready to publish by hitting "Run".
- Tryan interactive Jupyter or Jupyterlab session.

Reproducibility PI Manifesto

1. I will teach my graduate students about reproducibility.
2. All our research code (and writing) is under version control.
3. We will always carry out verification and validation (V&V reports are posted to figshare)
4. For main results in a paper, we will share data, plotting script & figure under CC-BY
5. We will upload the preprint to arXiv at the time of submission of a paper.
6. We will release code at the time of submission of a paper.
7. We will add a "Reproducibility" declaration at the end of each paper.
8. I will keep an up-to-date web presence.

Lorena A. Barba. 2012. Reproducibility PI Manifesto.
DOI:https://doi.org/10.6084/m9.figshare.104539.v1

Demo:

- **Reproducible packages of work from the Global Event and Trend Archive Research (GETAR) project:**
    - TwiRole
    - Event Focused Crawler

TwiRole capsule: https://codeocean.com/capsule/9584745/

Event Focused Crawler capsule: https://codeocean.com/capsule/8475497/

Thank you for your time :)

William A. Ingram and Edward A. Fox
Virginia Tech
Blacksburg, VA 24061
USA

{waingram, fox}@vt.edu

https://bit.ly/jcdl-reproducibility-pptx

References

- Artifact Review and Badging. Retrieved August 1, 2020 from https://www.acm.org/publications/policies/artifact-review-badging
- Monya Baker. 2016. 1,500 scientists lift the lid on reproducibility. Nature News 533, 7604 (May 2016), 452. DOI:https://doi.org/10.1038/533452a
- Lorena A. Barba. 2012. Reproducibility PI Manifesto. DOI:https://doi.org/10.6084/m9.figshare.104539.v1
- Carl Boettiger. 2015. An introduction to Docker for reproducible research. SIGOPS Oper. Syst. Rev. 49, 1 (January 2015), 71–79. DOI:https://doi.org/10.1145/2723872.2723882
- Karl Broman. 2016. Initial steps toward reproducible research. http://kbroman. org/steps2rr/.
- Jonathan B Buckheit and David L Donoho. 1995. Wavelab and reproducible research. In Wavelets and statistics. Springer, 55–81.
- Jon Claerbout. Making Scientific Contributions Reproducible. Retrieved July 29, 2020 from http://sepwww.stanford.edu/oldsep/matt/join/redoc/web/iris.html
- Jon F Claerbout and Martin Karrenbach. 1992. Electronic documents give reproducible research a new meaning. In SEG technical program expanded abstracts 1992. Society of Exploration Geophysicists, 601–604.
- Dav Clark, Aaron Culich, Brian Hamlin, and Ryan Lovett. 2014. BCE: Berkeley's Common Scientific Compute Environment for Research and Education. Austin, Texas, 5–12. DOI:https://doi.org/10.25080/Majora-14bd3278-002.

References

- David L. Donoho, Arian Maleki, Inam Ur Rahman, Morteza Shahram, and Victoria Stodden. 2009. Reproducible Research in Computational Harmonic Analysis. Computing in Science Engineering 11, 1 (January 2009), 8–18. DOI:https://doi.org/10.1109/MCSE.2009.15
- Jeff Hale. 2019. Learn Enough Docker to be Useful. Medium. Retrieved July 29, 2020 from https://towardsdatascience.com/learn-enough-docker-to-be-useful-b7ba70caeb4b
- Justin Kitzes, Daniel Turek, and Fatma Deniz (Eds.). 2017. The practice of reproducible research: case studies and lessons from the data-intensive sciences. Univ of California Press. Retrieved from https://www.practicereproducibleresearch.org/
- Donald E Knuth. 1992. Literate programming. Number 27 in CSLI lecture notes. Center for the Study of Language and Information (1992), 349–358.
- Olivier Mesnard and Lorena A. Barba. 2020. Reproducible Workflow on a Public Cloud for Computational Fluid Dynamics. Computing in Science Engineering 22, 1 (January 2020), 102–116. DOI:https://doi.org/10.1109/MCSE.2019.2941702 http://arxiv.org/abs/1904.07981
- Roger Peng. 2015. Report writing for data science in R. Leanpub. Retrieved from https://leanpub.com/reportwriting
- Roger D. Peng. 2011. Reproducible Research in Computational Science. Science 334, 6060 (December 2011), 1226–1227. DOI:https://doi.org/10.1126/science.1213847
- Jeffrey M. Perkel. 2019. Make code accessible with these cloud services. Nature 575, 7781 (November 2019), 247–248. DOI:https://doi.org/10.1038/d41586-019-03366-x

References

- Victoria Stodden, Friedrich Leisch, and Roger D. Peng. 2014. Implementing Reproducible Research. Chapman and Hall/CRC. DOI:https://doi.org/10.1201/9781315373461
- Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J. G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A. C. 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. 2016. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data 3, 1 (March 2016), 160018. DOI:https://doi.org/10.1038/sdata.2016.18
- Mark Woodbridge. Jupyter Notebooks and reproducible data science. Retrieved July 29, 2020 from https://markwoodbridge.com/2017/03/05/jupyter-reproducible-science.html
- Digital Preservation Management Workshops and Tutorial | Digital Preservation Management. Retrieved July 29, 2020 from https://dpworkshop.org/dpm-eng