Reproducible research

Good practices and useful information

G. Durif (CNRS – IMAG – Univ Montpellier)

Charles Elie Rabier (Univ Montpellier – IMAG)

Christelle Pierkot (CNRS – IR Data Terra)

Sonia Tieo (CNRS – CEFE – Univ Montpellier)

June 23th 2021, Montpellier

Montpellier Bio-Stats (`https://groupes.renater.fr/wiki/montpellier-biostat`)

Outline

1. Introduction
2. In the lab (no computations)
3. Reproducibility with computers
    - Software and programing
    - Software environment
4. Data
5. Scientific publication
6. Conclusion

Introduction

Resources

Desquilbet, L., Granger, S., Hejblum, B., Legrand, A., Pernot, P., Rougier, N.P., de Castro Guerra, E., Courbin-Coulaud, M., Duvaux, L., Gravier, P., Le Campion, G., Roux, S., Santos, F., 2019. **Vers une recherche reproductible.** Unité régionale de formation à l'information scientifique et technique de Bordeaux. [1]

The Turing Way Community, Becky Arnold, Louise Bowler, Sarah Gibson, Patricia Herterich, Rosie Higman, ... Kirstie Whitaker. (2019, March 25). **The Turing Way: A Handbook for Reproducible Data Science** (Version v0.0.4). Zenodo. [2]

Hejblum, B.P., Kunzmann, K., Lavagnini, E., Hutchinson, A., Robertson, D., Jones, S., Eckes-Shephard, A., 2020. **Realistic and Robust Reproducible Research for Biostatistics.** [3]

[1] https://hal.archives-ouvertes.fr/hal-02144142 and https://github.com/rr-france/bookrr

[2] http://doi.org/10.5281/zenodo.3233986 and https://github.com/alan-turing-institute/the-turing-way

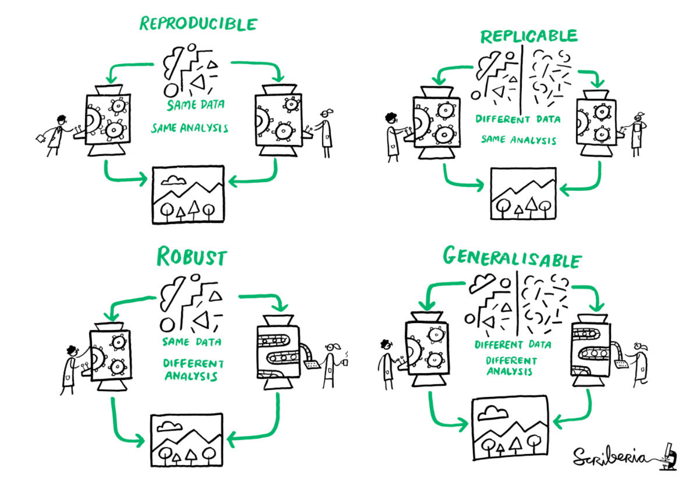
[3] https://doi.org/10.20944/preprints202006.0002.v1 and https://hal.inria.fr/hal-03100421

Reproducible research

- Many definitions...

- "A way of doing science so that scientific experiments, discoveries, results, etc. can be easily reproduced (done again), to be confirmed, or to be built on for the next study."

Reproducibility, replicability, robustness, generalization



Ref: The Turing Way Community and Scriberia (2019)

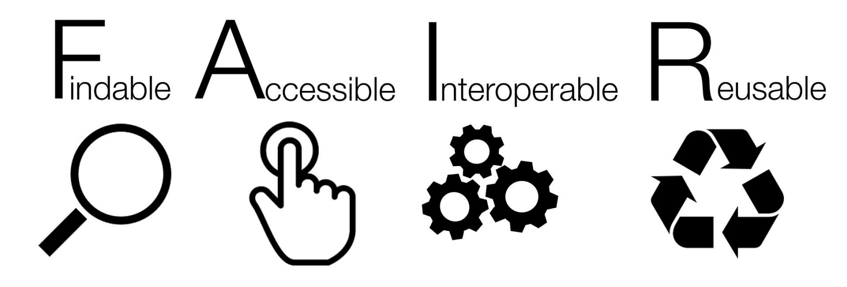
Different kind of reproducibility (for different kind of sciences)

- experimental reproducibility (without computation, at lab bench)
- reproducibility with computers
  - experimental reproducibility (“how to get similar results?”)
  - statistical reproducibility (“how to control randomness?”)
  - computational reproducibility (“how to get the exact same results?”)

→ more and more scientific results depends on some computer data processing (era of “computational” sciences)

Data

- "Open as much as possible and close as much as necessary"
- Management, publication, annotation (metadata), archiving

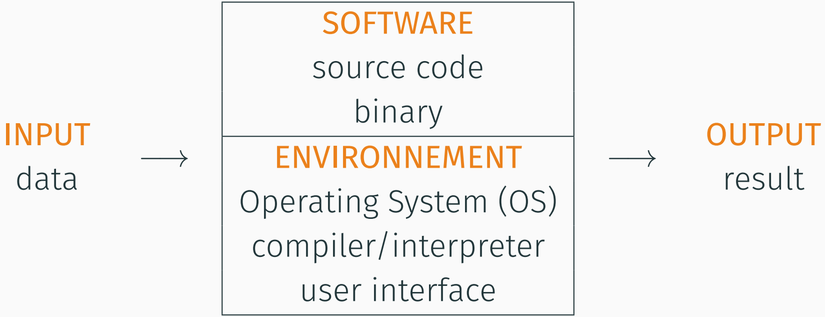


By SangyaPundir - Own work, CC BY-SA 4.0, `https://commons.wikimedia.org/w/index.php?curid=53414062`

- Source code = specific data (with specific consideration, c.f. later)

Software

- Computations are done by processing data through a software
- To run a software: you need a source code (or a binary) and an environment



INPUT

data

→

SOFTWARE

source code

binary

ENVIRONNEMENT

Operating System (OS)

compiler/interpreter

user interface

→

OUTPUT

result

Reproducible research = a requirement...

An increasing requirement for the scientist

- to publish (more and more scientific journals require sources to reproduce published results)
- to get financing and grants (sometimes)
- etc.

…but also (and almost) a good practice

A good practice to be adopted

- to make your life easier (to avoid the famous “how did I do that five days/weeks/months/years ago?”)
- to do quality research work (and avoid errors or frauds)
- to do incremental research (that can be used and built on in the future)

Open science

- “Movement to make scientific research (including publications, data, physical samples, and software) and its dissemination accessible to all levels of an inquiring society, amateur or professional” (Wiki, 2021c)[4]
- French comity for open science: `https://www.ouvrirlascience.fr`

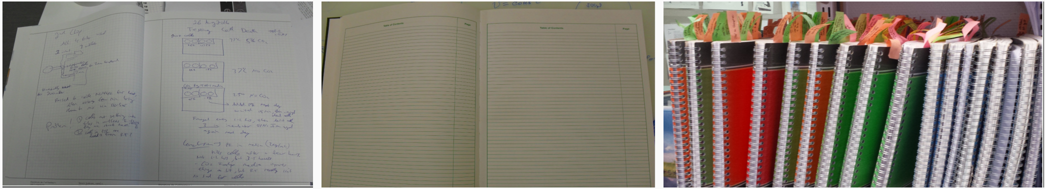
[4] `https://en.wikipedia.org/wiki/Open_science`

In the lab (no computations)

The lab notebook: good practice to log every experiments

Paper version:

- unreadable?
- missing or empty?
- when correctly filled: no index to find information (and no `CTRL+F`)



From eLabFTW (`https://www.elabftw.net`) presentation by Nicolas Carpi (Curie Institute, France)

Electronic lab notebook (ELN) software

- Experiment **description/annotation** and **metadata** (including data file, source code, machine configuration, etc.)
- **Timestamping** (registration of experiment date and time)
- **Export** to text/pdf/etc. (for readability, publication, archiving, etc.)
- **Legal issue**: mecanism to authenticate results and prevent falsification? (e.g. to proove anteriority)
- Proprietary/commercial solutions vs open source software ?

Resources regarding ELN

- Survey regarding ELN at CNRS (Léon and Libri, 2020)
- Open-source example: `elabFTW`[5] (CARPi et al., 2017)
- Meta-study (Kanza et al., 2017)
- Use case study (Oleksik et al., 2014)
- (Fairly) complete list (Huchet, 2021, webpage)

[5] `https://www.elabftw.net`

Reproducibility with computers

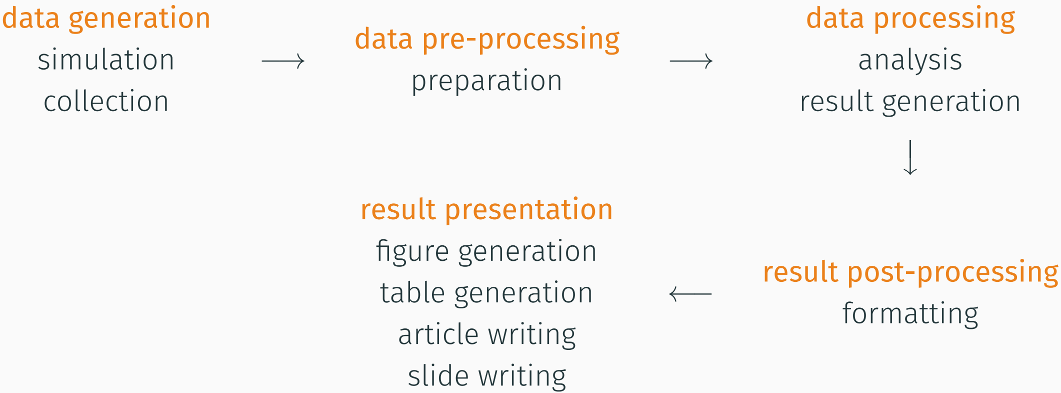
Reproducibility with computers (glossary may vary)

experimental reproducibility	similar input (data) + similar experimental protocol	→	similar results
statistical reproducibility	same input (data) + same analysis	→	same conclusions[6]
computational reproducibility	similar input (data) + same code/software + same software environment	→	exact same results[7]

[6] independently from (random) sampling variability

[7] bit-wise, i.e. bit-by-bit similarity

Experimental reproducibility with computers



data generation
simulation
collection

→

data pre-processing
preparation

→

data processing
analysis
result generation

↓

result post-processing
formatting

←

result presentation
figure generation
table generation
article writing
slide writing

Experimental reproducibility with computers

- Requirements: detailed experimental protocol, including all **data generation process**, **data pre-processing**, **data processing** (i.e. analysis) and **result post-processing**
- Good practice: **publish** the **source code** (e.g. scripts, notebooks, etc.) for your **entire analysis**[8] pipeline from data preparation to result formatting (including figures generation)

[8] and not just the source code of the methods/approaches that you developed

Statistical analysis and statistical reproducibility

Careful with common bad practices

- **Data manipulation/tempering** (justified or not) without explanation
  - selecting/removing datasets where your method performs well/poorly
  - removing observations of a dataset to improve results
- **Method "over-fitting"** on test/validation samples
- **Unexplained** parameter or hyper-parameter **calibration/tuning**
- **Over-trusting p-values** and **test result significance** without **controlling** the test **power** (why $\alpha = 5\%$ not 4.8% nor 5.2% ?)
- **Not accounting** for **confounding factors** or **hidden effects** (use randomization, blind control, sensitivity analysis, etc.)

Computational reproducibility

Requirements:

- input data
- source codes
- detailed software environment[9] (with corresponding versions)

WHY?[10]

- to detect mistakes or bugs more easily
- to understand and support/trust surprising or cutting-edge results
- to facilitate evolution and improvements[11]

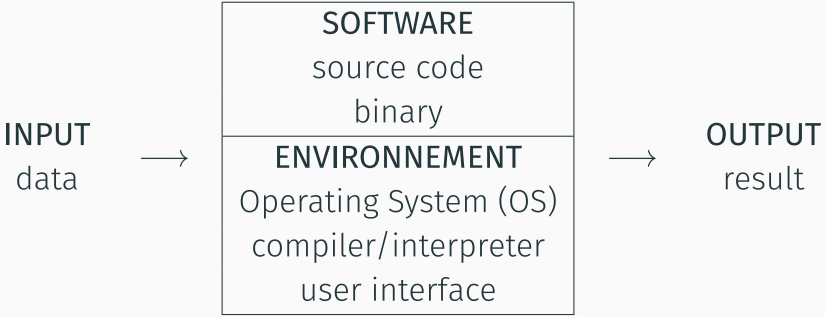
[9] which compiler was used to compile your binary program/interpreter/compiler?

[10] "Why experimental or at least statistical reproducibility is not enough????"

[11] especially when you stop working on the subject or maintaining the project

Computational reproducibility

This entire pipeline should be **reproducible** in a **deterministic**[9] way to achieve computational reproducibility.



INPUT

data

→

SOFTWARE

source code

binary

ENVIRONNEMENT

Operating System (OS)

compiler/interpreter

user interface

→

OUTPUT

result

[9] keep and store seeds when simulating random data

Software and programing (writing codes)

Choose a licence for your codes/softwares

- It governs the possibility to use, modify or redistribute a software
- It helps to identify clear authorship/copyright[10]
- Without a license: fuzzy and unclear (generally “all rights reserved” but you are never sure[11])
- **Recommandation:** use a free[12] and open-source license
- use a software specific license[13]

[10] depending on legal consideration, varying from one country to another

[11] “Was it forgotten or a deliberate choice?”

[12] as in “free” and not as in “gratis” (proprietary software can be gratis)

[13] e.g. Creative Commons lciense (`https://creativecommons.org/licenses/`) are for contents not softwares

Why a free and open-source (FOSS) licence?

WHY?

- your code/software is available for the community to use it, to improve it, to redistribute it[14]
- your code/software can be more easily used in other research works
- you cannot[15] be sure of what a proprietary closed software really does
- a good practice for open science and reproducible research
- a recommendation/obligation for publicly funded research work in France (Gruson-Daniel and Jean, 2021)

Note: you do not lose your authorship

[14] e.g. when your project ends and you stop maintaining your code/software

[15] at least not without huge difficulties

Why a free and open-source (FOSS) licence?

Different types of FOSS license[14] (see Laurent, 2004)

- permissive (MIT, Apache, BSD, etc.)
- copyleft (GPL, etc.)

Resources

- `https://choosealicense.com/`
- `https://opensource.org/licenses`
- `https://www.gnu.org/licenses/license-list.en.html`
- Distinction “free” vs “open source” (Stallman, 2009)

[14] The choice depends on your philosophy, your code/software purpose and user target audience

Good practice for software development and programming

- The code should be **human readable**[15] and **easily understandable** (use comments, code presentation and formatting)

  > Experiment: read your (5 weeks/months/years) old codes, are you sure that you will understand it? (worst with code written by others)

- Use a **versioning system** (e.g. `git`[16]) to manage your code evolution/version and for collaborative development

[15] being machine readable is necessary for the code to work but not sufficient
[16] Ref: `https://git-scm.com/book/`

Good practice for software development and programming

- Implement **automatic tests**[15] (e.g. unit tests) for each new function/module/etc. (and not afterward) to **verify your implementation and results** and avoid breaking your code[16]

- Write a **documentation**[17] for your code/package/library

[15] almost all programming languages offer testing functionality natively or in dedicated library (e.g. `testthat` in R, `pytest` in Python)

[16] never trust yourself, you will implement bugs

[17] almost all programming languages offer inline code documentation functionality natively or in dedicated library (e.g. `roxygen2` in R, `docstring` in Python)

Good practice for software development and programming

- Publish your source codes (preferably on a software forge)

- Archive your source codes (because your software forge or webpage can disappear[15])

References: Leprevost et al. (2014), Foord (2017), Coding best practices (Wiki, 2021a)

[15] See Agata et al. (2014) for instance

Software forge

An online server and/or website offering code/software development and management functionalities

- versioning
- collaborative work and planning
- issue, feedback, bug reports
- software release/publication
- continuous integration
- possibility to get a publication identification like a DOI[16]
- etc.

[16] eventually externally with `https://zenodo.org/`

Examples of software forge

- `gitlab`: free and open-source `git` forge hosting software (different hosts are available: in the academic world, e.g. `https://plmlab.math.cnrs.fr`, `https://gitlab.inria.fr`, or abroad, e.g. `https:gitlab.com`)
- `https://github.com`: very popular `git` forge with gratis and commercial solutions to host development projects
- `https://bitbucket.org`: another `git` forge with gratis and commercial solutions to host development projects

Discontinued[17] forges: Gitorious, Google code, Inria Gforge

[17] **Disclaimer:** it happens!

Publication $\neq$ archiving

- What happens if your software forge (or the webpage where you host your code) disappear ?

- The **Software Heritage** initiative (`https://www.softwareheritage.org`)

  > "Our ambition is to collect, preserve, and share all software that is publicly available in source code form. On this foundation, a wealth of applications can be built, ranging from cultural heritage to industry and research."

  → Simple deposit procedure from a software forge[18]

[18] `https://archive.softwareheritage.org/save/`

Code showcases/demos and result formatting/presentation

Recommendations: use a text file-based system[19]

- **Documented code scripts**
- **Raw text with formatting markup** (Markdown, LaTeX, etc.): readable even without the formatting software, exportable in different format
- **Literate programming** (Knuth, 1984): executable code chunks along with additional formatted text contents and explanations, like **notebooks** or **Org-mode**

[19] opening Office or PDF files can be a problem in the future, because of version conflict, discontinued software, etc.

Notebook

- Requirement: an interpreter like `jupyter` (`https://jupyter.org/`)
- Ideal to present results, figure/graph generation, code demos

Notebook

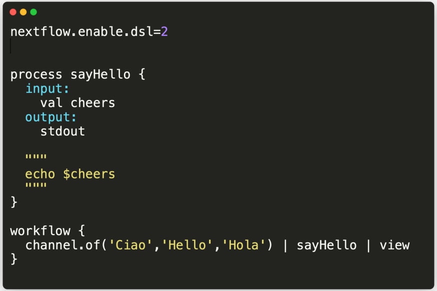
Limits:

- Suitable/convenient to run (heavy) computations[20]?
- Limited readability without the interpreter[21]: `json` based text format not easily readable in raw form if problem with interpreter

[20] compared to scripts

[21] compared to alternative like Markdown, Org-mode

Workflow system



```
nextflow.enable.dsl=2

process sayHello {
  input:
    val cheers
  output:
    stdout

  """
  echo $cheers
  """
}

workflow {
  channel.of('Ciao','Hello','Hola') | sayHello | view
}
```

Ref: `https://www.nextflow.io/`

Describe your complete workflow analysis with elementary bricks

Example: nextflow[22], snakemake[23], etc.

[22] `https://www.nextflow.io/`

[23] `https://snakemake.readthedocs.io/en/stable/`

Writing scientific material

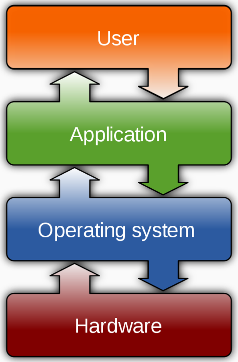
- **Final rendering of results (figures, tables, article, presentation) should also be reproducible!**
- Problem with ”what you see is what you get” tools like the Office Suite or alternatives (the information is lost without the software, potentially proprietary)
- **Writing with markup languages** (e.g. LaTeX or Markdown): content is readable and editable even without the rendering

Software environment
(and how to control it)

What is it?

The detailed description of the entire software stack (versions, availability) that is necessary to run a code/software

- Operating System (OS)
- Compiler and/or Interpreter (including the options used to compile/run the code)
- Additional libraries, external packages
- Hardware architecture on which the code was run (or can be run)



Ref: `https://commons.wikimedia.org/wiki/File:Operating_system_placement.svg`

Why is it necessary to control it?

- Programming languages[24], library implementations, Operating Systems (OS) **evolve**
- Potential **retro-compatibility issues** (e.g. try to run old `R` or `Python` codes with recent interpreters, or compile old codes with recent compilers)
- **Different implementations for standard operations** (e.g. the different implementations for pseudo-random number generators, or for the linear algebra librarie BLAS[25]: OpenBLAS, Atlas, Intel MKL, etc.)
- “What compiler was used to compile your compiler?”

[24] `R` 2.x.x, 3.x.x, 4.x.x, `Python` 2.x.x, 3.x.x, `C++` 11, 13, 17, 20, etc.

[25] used by `R`, `Numpy`

How to control your software environment?

- Describing your entire software and hardware stack? → cumbersome
- Container system (e.g. Docker, Singularity)
- Package manager system
- other?

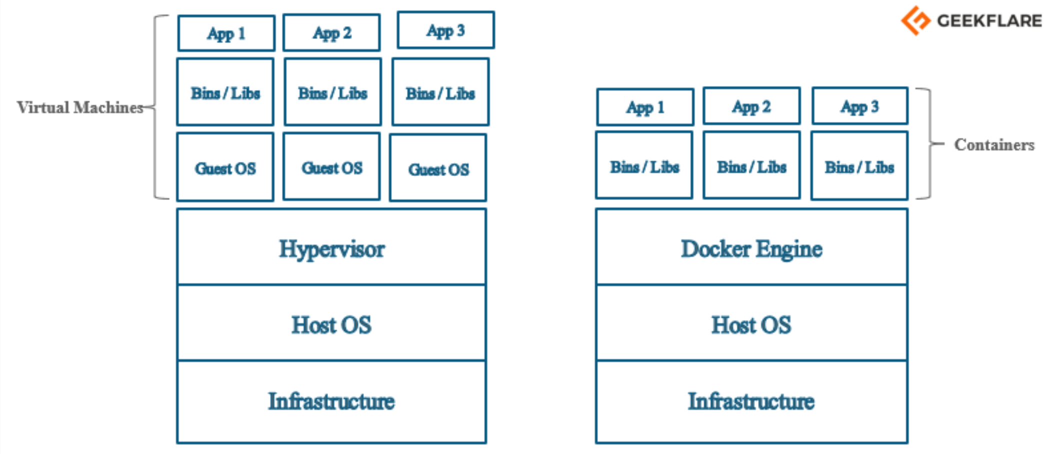
Container

- Operating-system-level virtualization
- Scriptable recipe to build executable versatile and configurable OS-like environments based on standard images, where you can run your programs
- Examples of systems: Docker[26], Singularity[27]

[26] https://www.docker.com/

[27] https://sylabs.io/singularity/

Container



Ref: `https://geekflare.com/fr/docker-vs-virtual-machine/`

Container

Advantages

- Easy **definition** and **control** of your software environment
- Possibility to **publish** (on your website, or on Docker/Singularity hubs) your containers so that other people can run your codes/programs in the same environment as you did (independently from their OS)

Limits

- Container **build** is generally **not reproducible** in a deterministic way
- Container recipe **rarely** follows **reproducible rules** and good practices.

Reproducible container?

```
FROM ubuntu:20.04
RUN apt-get update
   && apt-get upgrade -y
   && apt-get install -y ...
...
```

Reproducible container?

- `ubuntu:20.04`: regularly modified image
- `apt-get update` and `apt-get install`: install current version of packages
- Good practices: choose a stable image (and the smallest possible, e.g. `alpine`), include only the necessary libraries (e.g. no graphics libs if not used), avoid system updates[26]

[26] Disclaimer: we are talking about using container for reproducible purpose. In other context (e.g. to provide a web service, up-to-date libs/softwares are mandatory

Package manager

- specific to a language
  - e.g. pip[27]/conda[28] for Python, CRAN[29]/Bioconductor[30] for R
  - limits: management of package version? hidden requirements? evolution of the language?
- for a complete system
  - e.g. Guix, Nix, Debian
  - Example: Guix[31] to generate reproducible image (bit-by-bit), that store the complete dependence graph with all software versions

[27] `https://docs.conda.io`

[28] `https://pypi.org/`

[29] `https://cran.r-project.org/`

[30] `https://www.bioconductor.org/`

[31] `https://guix.gnu.org/`

Note on proprietary compilers/libraries

- GPU (Graphical Processing Units) computing: CUDA library for Nvidia GPUs[32], used by `PyTorch`, `TensorFlow`
- Intel compilers (ICC) and algebra library (MKL)

→ **fast** computations vs **reproducible** computations?

[32] trending in the machine learning community and elsewhere

Data

Ressources

- EOSC: `https://eosc-portal.eu`
- RDA: `https://rd-alliance.org`
- Certified data repositories: `https://www.coretrustseal.org/why-certification/certified-repositories/`
- *Comité "Ouvrir la Science" (CoSO)*: `https://www.ouvrirlascience.fr`

Why managing your data?

- Data generation: accumulation over the years
- For yourself and for others[33] (important for reproducibility)
- Data format: `https://facile.cines.fr`

[33] What happen to your scientific data when your project is over?

How?

- Data management plan (PGD)[34]
- Data repositories[35]

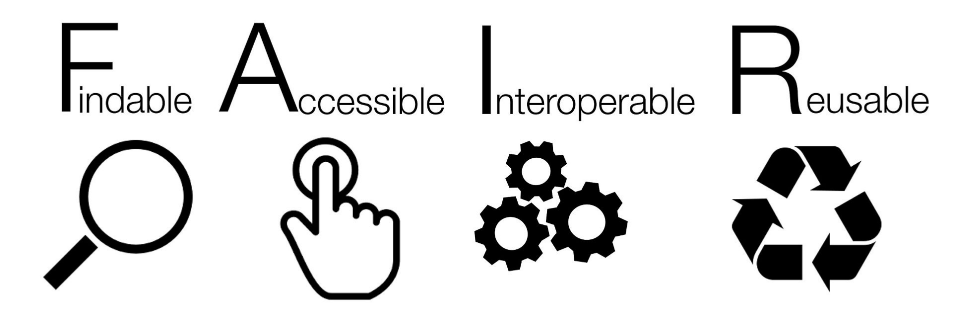
[34] https://www.ouvrirlascience.fr/plan-de-gestion-de-donnees-recommandations-a-lanr/

[35] https://hal.archives-ouvertes.fr/hal-02928817

Preserve and share

- value of your data?
- how data are collected/generated?
- time for data availability and duration of conservation?
- sharing with who? under which license? (share as much as possible, close as much as necessary)
- data cost: economical and environmental

FAIR principles[36]



By SangyaPundir - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=53414062

[36] See https://www.go-fair.org/fair-principles/ and https://teamopendata.org/t/open-data-et-fair-deux-paradigmes-differents/220

Publication ≠ archiving

- Publication: make your data accessible to the community
- Archiving: ensure your long-term data durability
- **Storing cost?** maybe OK during the project, but after? how to finance it?

Scientific publication

Open access

- “a set of principles and a range of practices through which research outputs are distributed online, free of cost or other access barriers” Wiki (2021b)[37]
- open access overview (Suber, 2007)
- open science principles (Swan, 2012, Unesco)
- publisher of open access journals: `https://www.openscience.fr`

[37] `https://fr.wikipedia.org/wiki/Libre_acc%C3%A8s_(%C3%A9dition_scientifique)` and `https://en.wikipedia.org/wiki/Open_access` (complementary)

HAL (`https://hal.archives-ouvertes.fr/`)

“HAL is an open archive where authors can deposit scholarly documents from all academic fields.”

- Open repository to upload and index any publication, preprint, etc., including metadata and contents
- Possible to define an embargo on the contents (that is indexed but not available for a given time)
- Multiple sub-repositories: Inria[38], INRAE[39], TEL[40] (PhD manuscripts)

[38] https://hal.inria.fr

[39] https://hal.inrae.fr

[40] https://tel.archives-ouvertes.fr

How do scientific journals address science’s reproducibility issues ?

A small tour of scientific journals …

PLOS journals (interdisciplinary journals)

PLOS publishes a suite of influential Open Access journals across all areas of science and medicine

PLOS ONE
A peer-reviewed, open access journal

PLOS | BIOLOGY

PLOS DIGITAL HEALTH

PLOS | COMPUTATIONAL BIOLOGY

PLOS | GENETICS

PLOS WATER

PLOS | MEDICINE

PLOS | NEGLECTED TROPICAL DISEASES

Publication fees :

Plos One (1749 dollars), Plos Genetics (2575 dollars), Plos Computational Biology (2575 dollars) . . .

PLOS journals (interdisciplinary journals)

“PLOS is committed to ensuring the availability of materials that underpin research. **Sharing materials** encourages reuse and **facilitates reproducibility**.”

“PLOS reserves the right to issue a correction, expression of concern, or retraction if unreasonable restrictions on sharing are discovered after publication.”

“**All data and related metadata** underlying the findings reported in a submitted manuscript should be deposited **in an appropriate public repository**”

DRYAD figshare zenodo

kaggle OPEN SCIENCE FRAMEWORK

PNAS and Genome Research (interdisciplinary journals)

PNAS
Proceedings of the
National Academy of Sciences
of the United States of America

“Authors must make materials, data, and associated protocols, including code and scripts, available to readers upon publication. Authors should **deposit data** in community-approved **public repositories** prior to publication”

GR

“**Genome Research will not consider manuscripts in which the data** used and reported in the paper that are **required for reproducibility are not freely available** in either a public database or on the Genome Research website”

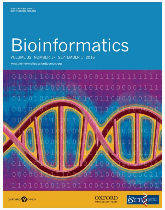
Nature journals (interdisciplinary journals)

nature

“**Nature** Portfolio journals **aim to improve** the transparency of reporting and **reproducibility of published results** across all areas of science”

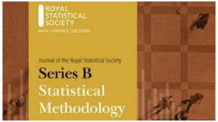
“A condition of publication in a Nature Portfolio journal is that **authors are required to make materials, data, code** and associated protocols promptly **available** to readers without undue qualifications”

Bioinformatics


Bioinformatics
VOLUME 32 NUMBER 17 SEPTEMBER 1 2016
www.bioinformatics.oxfordjournals.org
OXFORD UNIVERSITY PRESS
ISCB

“Bioinformatics is aligned with the general movement towards open FAIR data. All data on which the conclusions given in the publication are based must be publicly available in stable public repositories.”

JRSSB (a statistical journal)


ROYAL
STATISTICAL
SOCIETY
DATA | EVIDENCE | DECISIONS
Journal of the Royal Statistical Society
Series B
Statistical
Methodology

"Published papers should, where possible, be accompanied by the data and computer code used in the analysis. Both data and code must be clearly and precisely documented, in enough detail that it is possible to replicate all results in the final version of the paper"

JASA (a statistical journal)

Journal of the
American
Statistical
Association

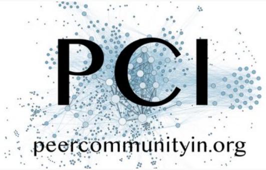
June 2021 ❥ Volume 116 ❥ Number 534

“To enhance the reproducibility of published research, **manuscripts undergo reproducibility review** …”

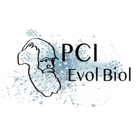
Reproducibility Review Form (JASA)

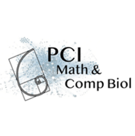
1. Data availability: data available in a public repository ?
2. Data integrity: data provided with the submission match with data originally available to the authors ?
3. Data documentation and usability
4. Code availability: code available in a public repository ?
5. Code clarity: code in a form that can be used and understood by others ?
6. Documentation of workflow: clear documented workflow (including data preparation/cleaning steps and analyses) to reproduce the results ?
7. Reproducibility
    - **without having run the code**, any concerns that the code would not reproduce the key results ?
    - **based on having run the code**, did the workflow allow you to reproduce the key results?

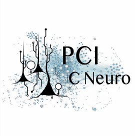
The journals “Peer Community in” (PCI)



The functioning of PCI[41]







[41] See the introduction video at `https://www.youtube.com/watch?v=4PZhpnc8wwo`

The journal “Rescience”

“Reproducible Science is good. Replicated Science is better”

- ReScience C = platinum open-access peer-reviewed journal (100% free)
- Explicit **replication of already published research**
- **New implementation** of a replicated computational results from the literature

Ten Years Reproducibility Challenge (special issue from 2020)

- Invitation for researchers to **try to run their old code** created for a publication ($\geq$ 10 years)
- **Try to make your old code work** on modern hardware/software in order to check if you obtain the same results

Conclusion

Take-home message

Reproducible research... a journey!

- necessary and **useful** to do incremental research, for others but also for yourself

- an investment: heavy need to change the behaviors and practices in science (regarding experiments, publications, management)

Message to senior researchers

- Change will come from the top (young researchers follow what is expected to advance in their career)
- Reproducible research is not compatible with publication race
- Improve scientific training and career management

Environmental questions

Environmental cost of computations, data storage?

Thank you for you attention

Questions?

`https://groupes.renater.fr/wiki/montpellier-biostat`

References

Agata, T., Y. Miyata, E. Ishita, A. Ikeuchi, and S. Ueda (2014, December). Life span of web pages: A survey of 10 million pages collected in 2001. *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, 463–464.

CARPi, N., A. Minges, and M. Piel (2017, April). eLabFTW: An open source laboratory notebook for research labs. *Journal of Open Source Software 2*(12), 146.

Foord, M. (2017, May). 30 best practices for software development and testing. https://opensource.com/article/17/5/30-best-practices-software-development-and-testing.

Gruson-Daniel, C. and B. Jean (2021, January). étude relative à l'ouverture des codes sources au sein de l'Enseignement Supérieur et de la

Recherche (ESR) : Considérations en termes d'usage et de valeur. Research Report, INNO3 ; Etalab ; Comité pour la Science Ouverte.

Huchet, B. (2021). 2021 Review of the Best Electronic Laboratory Notebooks | Labs Explorer. https://www.labsexplorer.com/c/2021-review-of-the-best-electronic-laboratory-notebooks_222.

Kanza, S., C. Willoughby, N. Gibbins, R. Whitby, J. G. Frey, J. Erjavec, K. Zupančič, M. Hren, and K. Kovač (2017, May). Electronic lab notebooks: Can they replace paper? *Journal of Cheminformatics 9*(1), 31.

Knuth, D. E. (1984). Literate programming. *The Computer Journal 27*(2), 97–111.

Laurent, A. M. S. (2004, August). *Understanding Open Source and Free Software Licensing: Guide to Navigating Licensing Issues in Existing & New Software*. "O'Reilly Media, Inc.".

Léon, N. and D. Libri (2020). Analyse de l'enquête sur les cahiers de laboratoire électroniques au CNRS. Technical report, CNRS.

Leprevost, F. d. V., V. C. Barbosa, E. L. Francisco, Y. Perez-Riverol, and P. C. Carvalho (2014, July). On best practices in the development of bioinformatics software. *Frontiers in Genetics 5*.

Oleksik, G., N. Milic-Frayling, and R. Jones (2014, February). Study of electronic lab notebook design and practices that emerged in a collaborative scientific environment. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, New York, NY, USA, pp. 120–133. Association for Computing Machinery.

Stallman, R. (2009, June). Viewpoint: Why "open source" misses the point of free software. *Communications of the ACM 52*(6), 31–33.

Suber, P. (2007). *Open Access Overview*.

Swan, A. (2012). *Policy Guidelines for the Development and Promotion of Open Access*. UNESCO.

The Turing Way Community and Scriberia (2019, July). Illustrations from the Turing Way book dashes.

Wiki (2021a, May). Coding best practices. *Wikipedia*.

Wiki (2021b, June). Open access. *Wikipedia*.

Wiki (2021c, June). Open science. *Wikipedia*.