FAIR_bioinfo for bioinformaticians

Introduction to the tools of reproducibility in bioinformatics

C. Hernandez[1] T. Denecker[1] J.Sellier[2] C. Toffano-Nioche[1]

[1]Institute for Integrative Biology of the Cell (I2BC)
UMR 9198, Université Paris-Sud, CNRS, CEA
91190 - Gif-sur-Yvette, France

[2]Institut de Génétique et de Biologie Moléculaire et Cellulaire (IGBMC)
CNRS UMR 7104 - Inserm U 1258
67404 - Illkirch cedex, France

Sept. 2020

Conclusion

Training schedule

Day 1:

- Introduction to FAIR_bioinfo
- Encapsulation (docker)
- Workflow (SNAKEMAKE)
- IFB resources (, slurm)

Day 2:

- History management (git, GitHub)
- Software environment management (CONDA)
- Traceability with notebooks (jupyter, rmarkdown)
- Sharing and disseminating (GitHub, zenodo)

FAIR_bioinfo

Let's take a step back.

FAIR_bioinfo

Findable

Easy to find protocols (GitHub GitHub Pages) with DOI (zenodo)

Accessible

Open source (GitHub, docker, CONDA, ...)

Interoperable

Think "workflow" (SNAKEMAKE + docker / CONDA) locally or on servers (, slurm)

Reusable

Replayable protocols (jupyter, markdown) in virtual environments (docker / CONDA)

A virtuous cycle

FAIR raw data

+

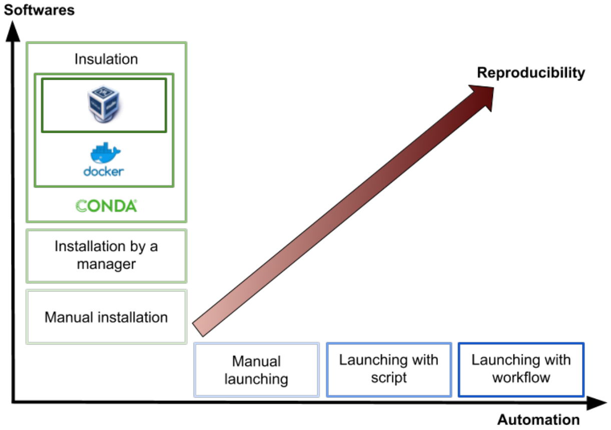
FAIR_bioinfo scripts/protocols

=

FAIR processed data

FAIR_bioinfo

Reproducibility is a multi-dimensional process



So... What now?

What now?

Automation

- Manual
- Write a script
- Use a workflow manager

Software

- Local installation
- Package manager
- Conda environment
- Image / container
- Virtual machine

Continuous integration

Verification at each source code modification that the result of the modifications does not produce:

- no regression in the developed application
- nor any change in the results obtained

Travis CI

circleci

GitHub Actions

Some FAIR_bioinfo limits

Usability, effort/cost & simplicity
Reproducibility to the exact bit

FAIR_bioinfo training

✗ use of an already instantiated VM
✓ create your own VM image

Reproducibility to the exact bit?

✗ container uses some resources of the support machine
✗ version control of the env. (Nix, Guix)

Parallelization:

✗ loss of computational order, multi-threading, same hardware?
✗ ...?

Reproducibility checklist[1]

- Code Enshrine computations and data manipulation in code, avoid workflows based on point-and-click interfaces (eg. Excel)
- Document Explain how code works, define parameters and computational environment required: comments, notebooks and README
- Record Note key parameters (eg. the 'seed' values of a random-number generator)
- Test with test functions using positive and negative control data sets, and run those tests throughout development
- Guide with master script (eg. 'run.sh') that downloads data sets and executes workflow
- Archive with long-term stability services such as Zenodo, Figshare and Software Heritage (GitHub is impermanent online repository).

[1] Nature

Reproducibility checklist[2]

- Track the project's history with a version-control tools (eg. Git). Note which version you used to create each result
- Package with ready-to-use computational environments using containerization tools (eg. Docker, Singularity), web services (Code Ocean, Gigantum, Binder) or virtual-environment managers (Conda)
- Automate the test of your code with continuous-integration services (eg. Travis CI)
- Simplify Avoid niche or hard-to-install third-party code libraries
- Verify your code's portability by running it in a range of computing environments

[2] Nature

Thanks

- Organizational comity (our guardian angels): Yousra, Jacques, Hélène
- NNCR team force: Julien, Gildas, and those who provide in the shadows
- beta-testers: Pauline
- Organisations: CNRS, INRAE, IFB, I2BC, Paris Saclay University, ...