FAIR_bioinfo : Open Science and FAIR principles in a bioinformatics project

How to make a bioinformatics project more reproducible

C. Hernandez[1] T. Denecker[2] J. Sellier[2] G. Le Corguillé[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] IFB Core Cluster taskforce

June 2021

Conclusion

Conclusion

Current schedule

Day 1:

- Introduction to FAIR_bioinfo
- History management (git, GitHub)
- Environment management (CONDA, docker)

Day 2:

- Workflow (SNAKEMAKE)
- Traceability with notebooks (jupyter, rmarkdown)
- IFB resources (S, slurm)
- 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

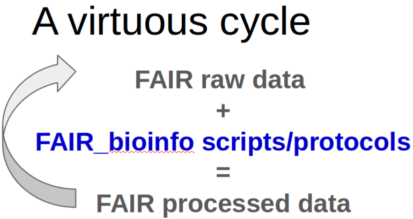
Open source (GitHub, docker, CONDA, ...)

Interoperable

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

Reusable

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



A virtuous cycle

FAIR raw data

+

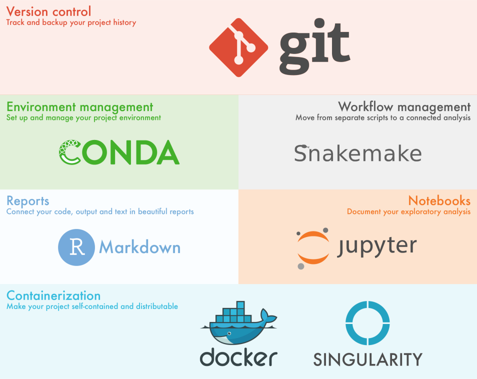
FAIR_bioinfo scripts/protocols

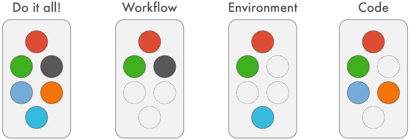
=

FAIR processed data

Swedish similar tutorial

From the NBIS – National Bioinformatics Infrastructure Sweden


Version control
Track and backup your project history
git
Environment management
Set up and manage your project environment
CONDA
Workflow management
Move from separate scripts to a connected analysis
Snakemake
Reports
Connect your code, output and text in beautiful reports
R Markdown
Notebooks
Document your exploratory analysis
jupyter
Containerization
Make your project self-contained and distributable
docker
SINGULARITY



nbis-reproducible-research.readthedocs.io/en/latest

Reproducibility checklist[1]

- Code avoid workflows based on point-and-click interfaces (eg. Excel), enshrine computations and data manipulation in code
- Document how code works, define parameters and computational environment required: comments, notebooks and README
- Record key parameters (eg. the 'seed' values of a random-number generator)
- Test functions using positive and negative control data sets, 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 (tag) 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)
- Simplify and avoid niche or hard-to-install third-party code libraries
- Verify your code's portability by running it in a range of computing environments
- Automate the test of your code with continuous-integration services (eg. Travis CI)

[2] Nature

Adding Tests

Unit test: test a part of the code

```r
## module 1
sum <- function(x, y){
    return (x+y)
}

# Unit test
sum(2,2) == 4
```

```r
## module 2
power <- function(x, y){
    return (x**y)
}

# Unit test
power(2,2) == 4
```

Functional test: test all the code

```r
# Functional test
power(sum(2,2),2) == 16
```

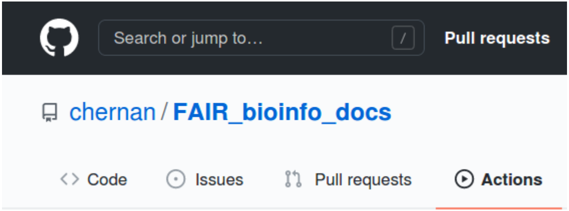
Continuous integration

Automated verification each time the source code is modified that the modifications do not produce:

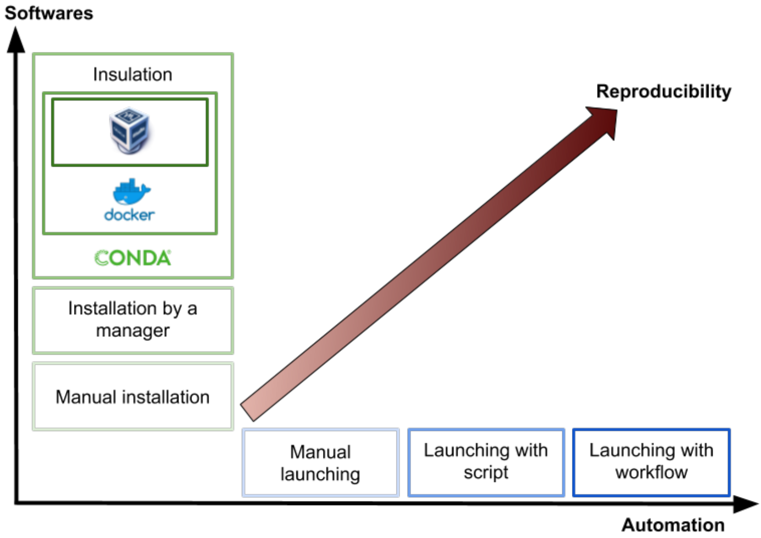
- any regression in the developed application
- any change in the results obtained

Travis CI

circleci



Reproducibility: a multidimensional and multi-level process



FAIR_bioinfo

Automation

Manual command lines

↓

Write a shell script

↓

Use a workflow manager

↓

Tests and continuous integration (*)

User analysis (trial-and-error)

Offer a GUI (eg. with R-Shiny) (*)

↓

Save and re-import choices (*)

Softwares

Local installation

↓

Package manager

↓

Conda environment

↓

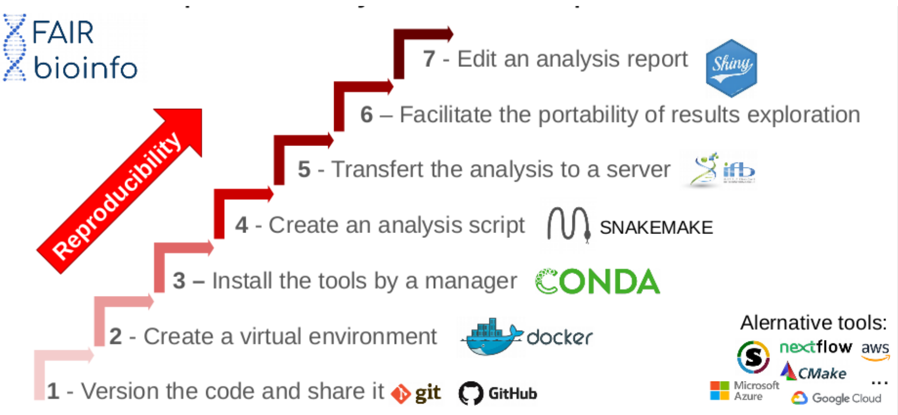
Image / container

↓

Virtual machine (*)

(*) not carried out in the course

FAIR_bioinfo



FAIR bioinfo

Reproducibility

7 - Edit an analysis report (Shiny)

6 – Facilitate the portability of results exploration

5 - Transfert the analysis to a server (ifb)

4 - Create an analysis script (SNAKEMAKE)

3 – Install the tools by a manager (CONDA)

2 - Create a virtual environment (docker)

1 - Version the code and share it (git, GitHub)

Alernative tools:

Reproducibility - how far?

Reproducibility to the exact bit?

✗ container uses some resources of the support machine

✓ version control of the env. (Nix, Guix)

HPC and parallelization?

✗ loss of computanional order, multi-threading, identical hardware?

✓ ...?

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

Thanks

- Organizational comity (our guardian angels): Yousra, Hélène
- IFB Core Cluster taskforce: Julien, Gildas, and all those who provide in the shadows
- Helpers: Paulette, Emilie, Pauline, Hugo
- Organisations: CNRS, INRAE, IFB, I2BC, Paris Saclay University