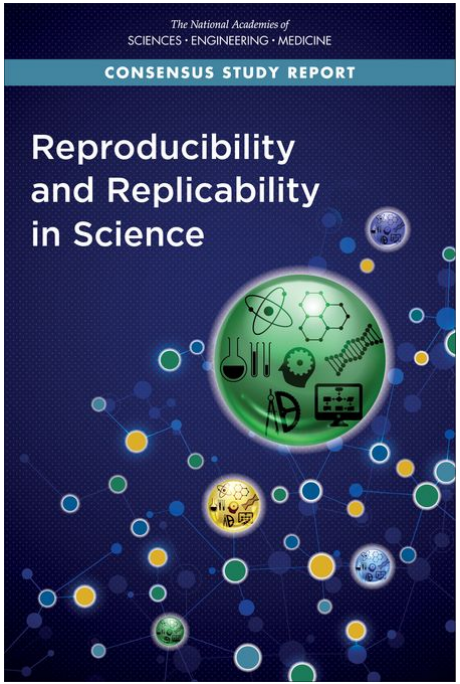
La reproductibilité au service de la Biologie Computationnelle

Thomas DENECKER

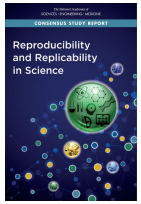
23 mai 2019

La reproductibilité en Sciences en 2019



National Academies of Sciences, Engineering, and Medicine. 2019. Reproducibility and Replicability in Science. Washington, DC: The National Academies Press. https://doi.org/10.17226/25303.

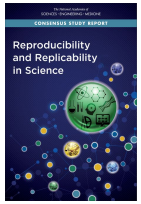
Qu’est-ce que la reproductibilité ?



Reproductibilité

Obtenir des résultats de calcul cohérents en utilisant les mêmes données d'entrée, étapes de calcul, méthodes, codes et conditions d'analyses.

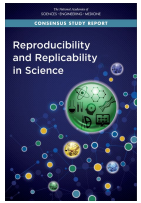
La reproductibilité *versus* la réplicabilité



Reproductibilité

Obtenir des résultats de calcul cohérents en utilisant les mêmes données d'entrée, étapes de calcul, méthodes, codes et conditions d'analyses.

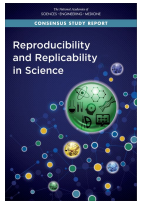
≠



Réplicabilité

Obtenir des résultats cohérents dans toutes les études visant à répondre à la même question scientifique, chacune ayant analysé ses propres données.

La reproductibilité *versus* la répétabilité



Reproductibilité

Obtenir des résultats de calcul cohérents en utilisant les mêmes données d'entrée, étapes de calcul, méthodes, codes et conditions d'analyses.

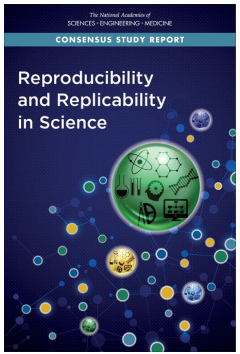
≠

Répétabilité (def. adaptée de Hans E. Plesser, *Front. Neuroinf.* , 2017)

Obtenir les résultats les plus proches possibles en refaisant une expérience à l'identique (méthodes, équipements, expérimentateurs, laboratoire et conditions)

frontiers in Neuroinformatics

Recommandations pour être reproductible en 2019



Description de la partie expérimentale

Méthodes, instruments, procédures, mesures, conditions expérimentales

Description de la partie computationnelle

Etapes de l'analyse des données et choix techniques

Description de la partie statistique

Décisions analytiques : quand, comment, pourquoi

Discussion des choix et des résultats obtenus

Dans la réalité ?

Un problème de reproductibilité en Biologie

70 %

des analyses en Biologie expérimentale ne sont **pas reproductibles**

(Monya Baker, *Nature*, 2016)

La bioinformatique est aussi touchée ...

Des problèmes encore trop fréquents

- **Impossibilité d'installer des outils**
  - OS non compatible
  - Dépendance plus disponible
- **Mise à jour de l'outil rendant inutilisable les codes**
  - Python 2 et Python 3 !
  - Changement des arguments des fonctions utilisées (R)
- **Impossibilité de reproduire les résultats de l'analyse computationnelle**
  - IDE : version stable du langage différente selon l'OS (Rstudio)
  - Version des packages

Il existe heureusement des solutions !

Quand la bioinformatique rencontre les technologies du développement

SNAKEMAKE

...

Un exemple de reproductibilité en Bioinformatique

Formation à l'I2BC avec Claire Toffano-Nioche

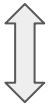
Exploiter les principes FAIR data pour rendre un protocole d'analyse reproductible et obtenir toujours les mêmes résultats à partir des mêmes données

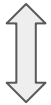
Tous les cours et les codes sont *open source*

https://github.com/thomasdenecker/FAIR_Bioinfo

Proposition d’une solution

Des équivalents pour chaque choix



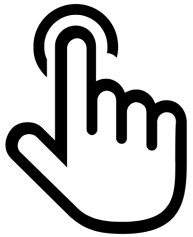


...

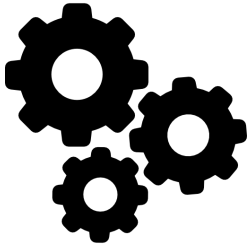
Les principes FAIR data

Findable

Accessible



Interoperable



Reusable

Comment exploiter les principes FAIR data?

F

A

I

R

Comment exploiter les principes FAIR data?

Facile à trouver

- Outils tiers utilisés = références dans leur domaine
- Protocole d'analyses simple à trouver (Github pages)

A

I

R

Comment exploiter les principes FAIR data?

Facile à trouver

- Outils tiers utilisés = références dans leur domaine
- Protocole d'analyses simple à trouver (Github pages)

Accessible

- Ressources disponibles (Github, dockerhub)
- Outils tiers *open source* (conda)

I

R

Comment exploiter les principes FAIR data?

Facile à trouver

- Outils tiers utilisés = références dans leur domaine
- Protocole d'analyses simple à trouver (Github pages)

Accessible

- Ressources disponibles (Github, dockerhub)
- Outils tiers *open source* (conda)

Interopérable

- Coopération des outils (snakemake, docker) aussi bien en local que sur serveurs (cloud ou cluster)

R

Comment exploiter les principes FAIR data?

Facile à trouver

- Outils tiers utilisés = références dans leur domaine
- Protocole d'analyses simple à trouver (Github pages)

Accessible

- Ressources disponibles (Github, dockerhub)
- Outils tiers *open source* (conda)

Interopérable

- Coopération des outils (snakemake, docker) aussi bien en local que sur serveurs (cloud ou cluster)

Réutilisable

- Protocole rejouable simplement (snakemake) à l'identique (Jupyter) dans un environnement virtuel (docker)

Notre crédo : So FAIR !

FAIR raw data

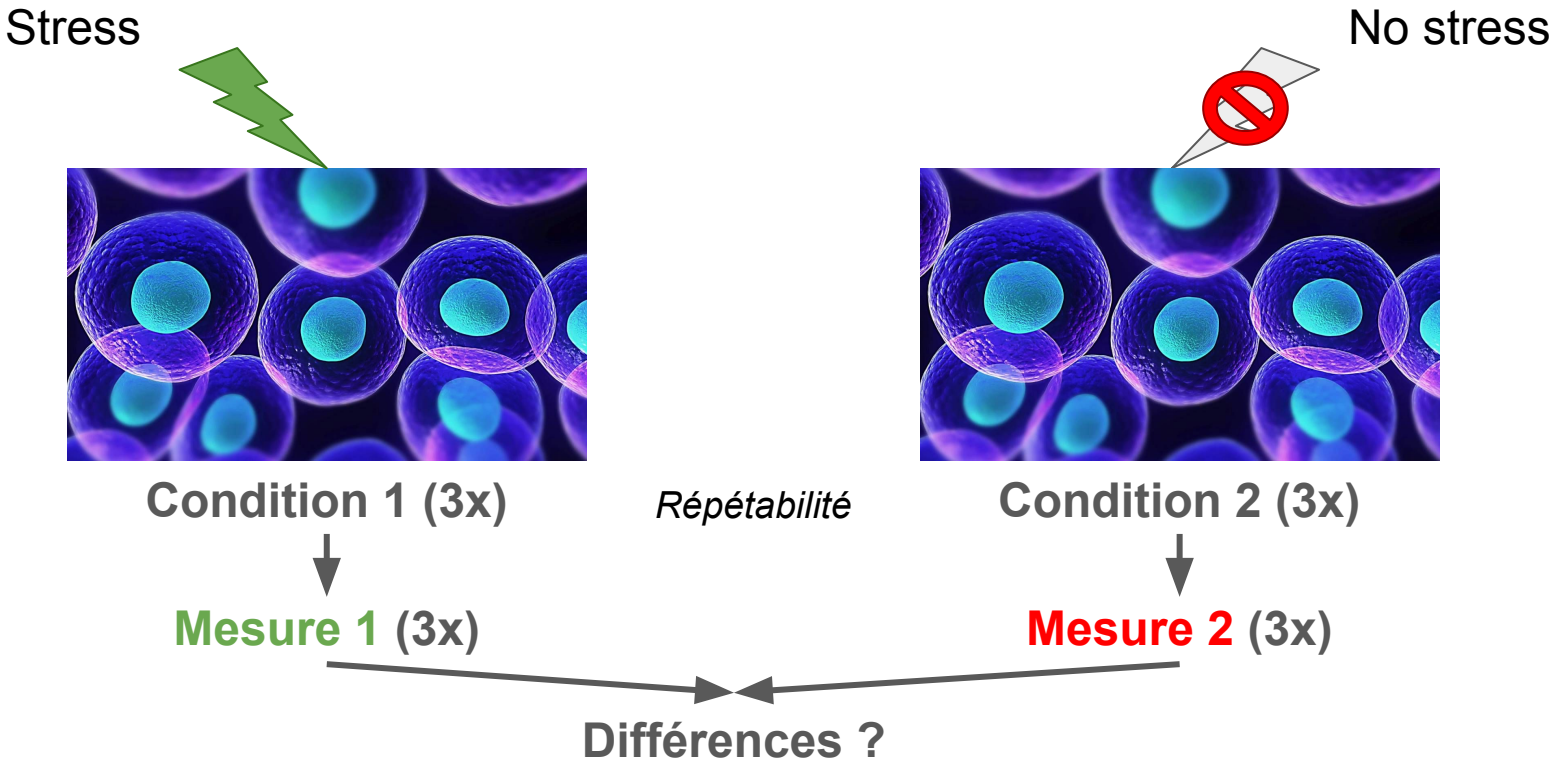
+

FAIR_bioinfo scripts/protocols

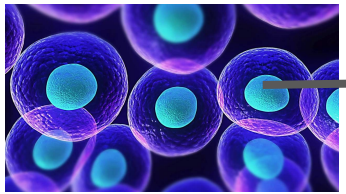
=

FAIR processed data

Les données : un peu de biologie ?

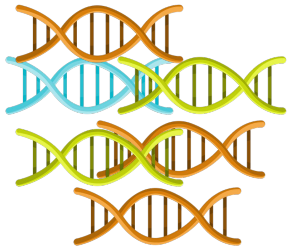


Niveau d’expression des gènes



Cellules dans une condition

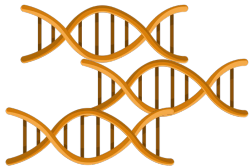
Mesure m



ARN → ADNc

Pipeline d’analyses

Gène 1



Gène 2

...

Gène n

Table de comptage

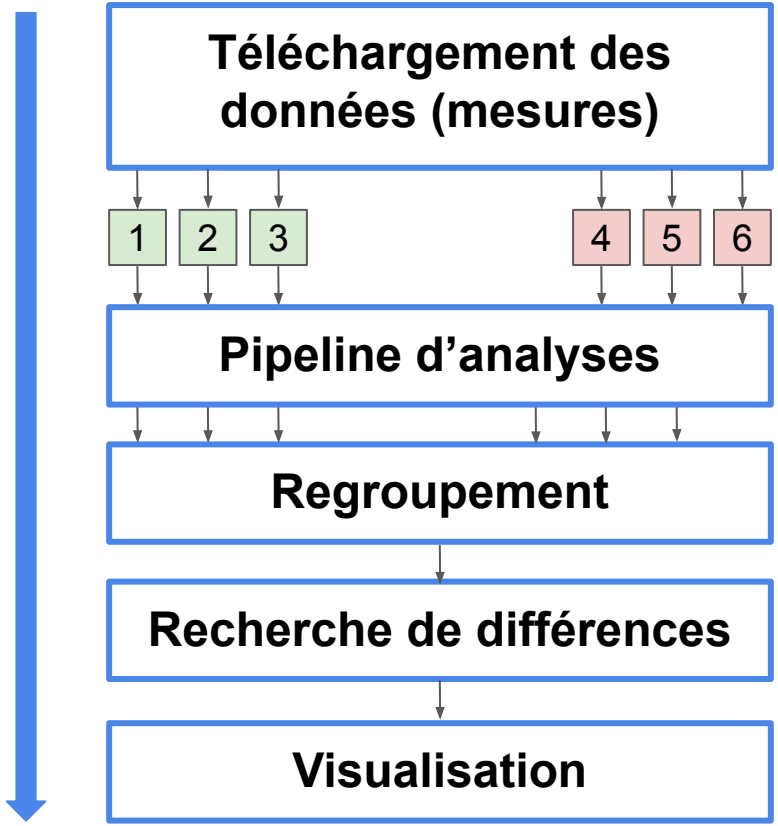
Gène 1	1
Gène 2	3
...	
Gène n	2

Différences entre les conditions

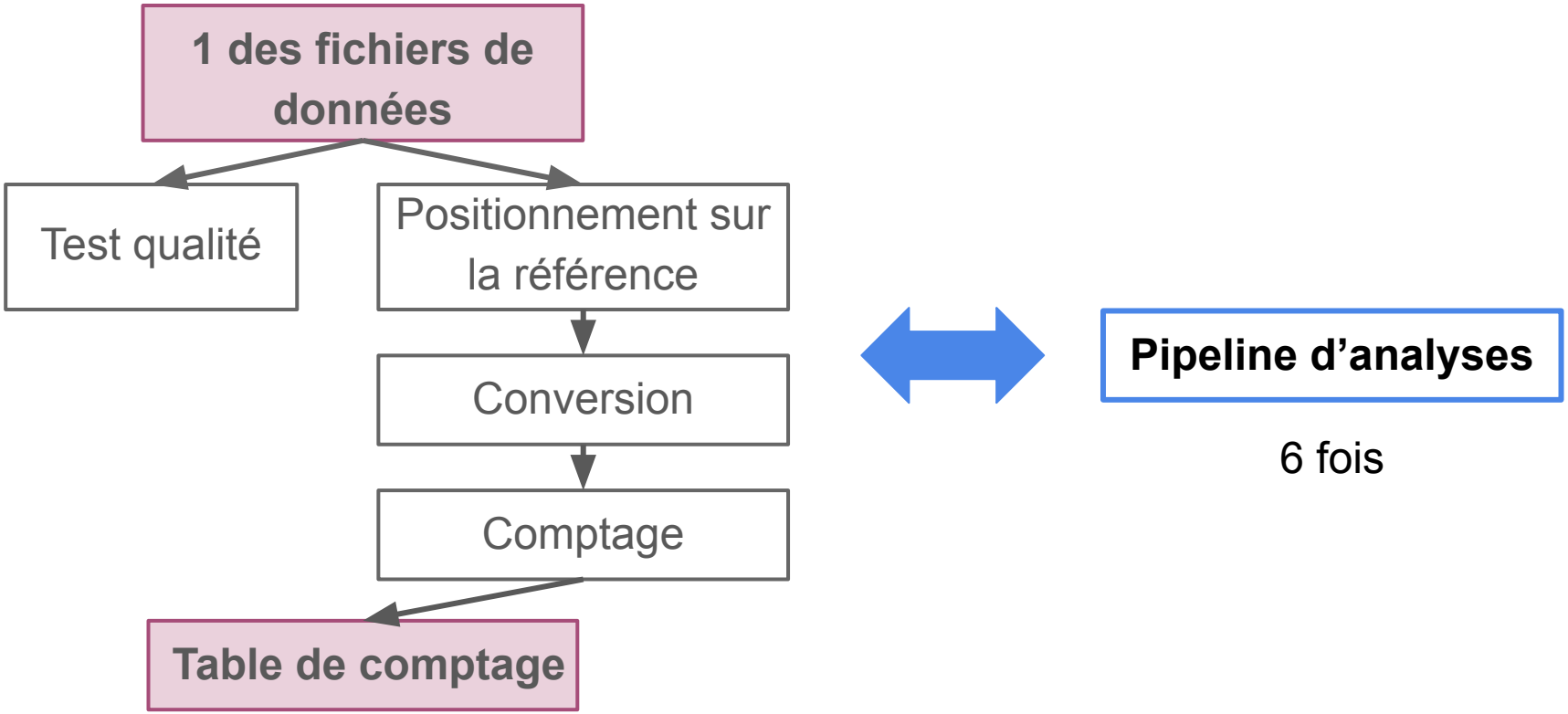
	Condition 1			Condition 2			
	1	**2**	**3**	**4**	**5**	**6**	
Gène 1	1	1	1	3	3	3	≠
Gène 2	3	3	3	1	1	1	≠
...	...	...	...	...	...	...	
Gène n	2	2	2	2	2	2	=

Conclusion
Gènes 1 et 2 différentiellement exprimés entre la condition 1 et 2

Comment traiter ces données ?



Pipeline d’analyses



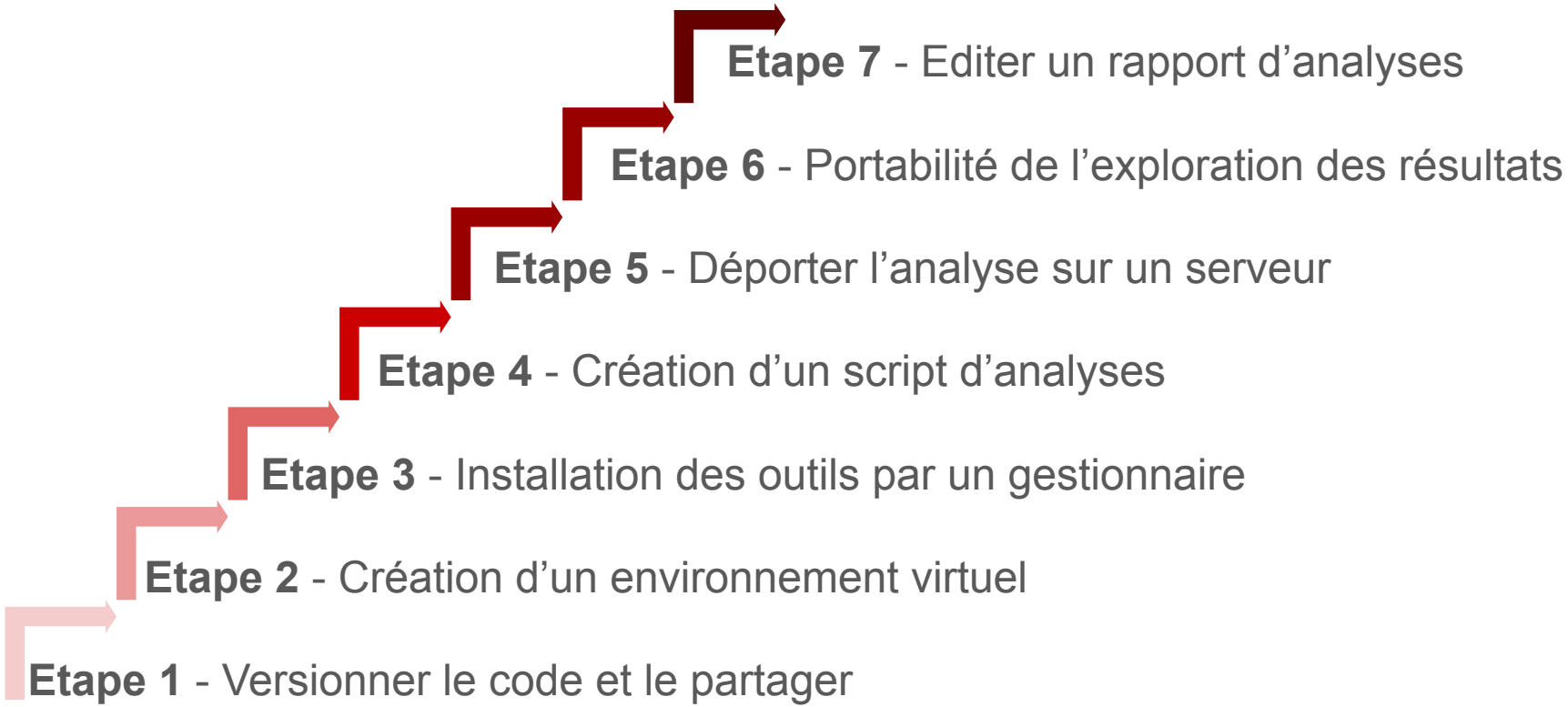
Ce qui se fait majoritairement

1. Installer les outils en local (parfois écriture d'un script d'installation)
2. Ecrire un script pour lancer toutes les analyses (pas toujours …)
3. Partage du script (par la publication, par mail, clé USB, ...)

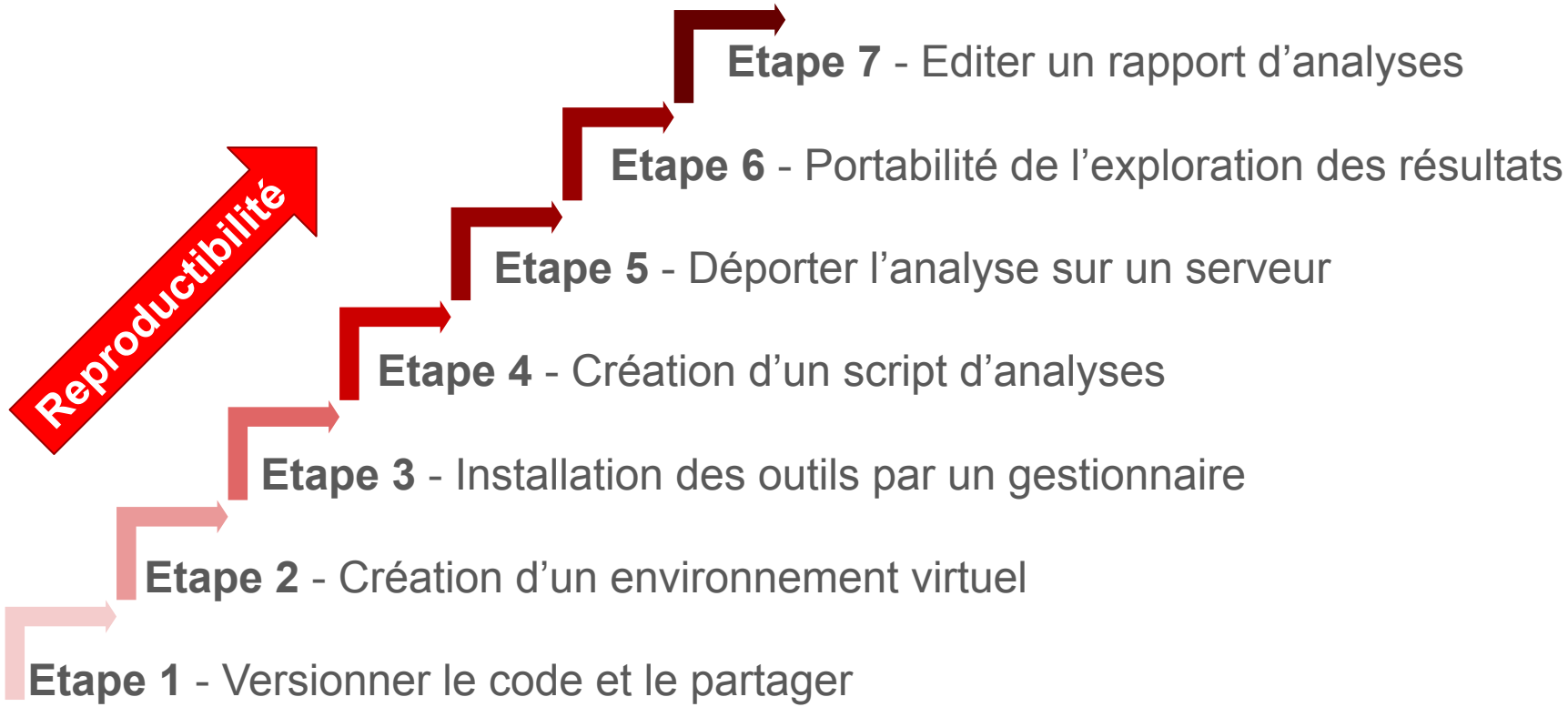
Mais peu de reproductibilité ...

Quelles sont les solutions pour être plus reproductible en Bioinformatique ?

Une solution en 7 étapes



Une solution en 7 étapes



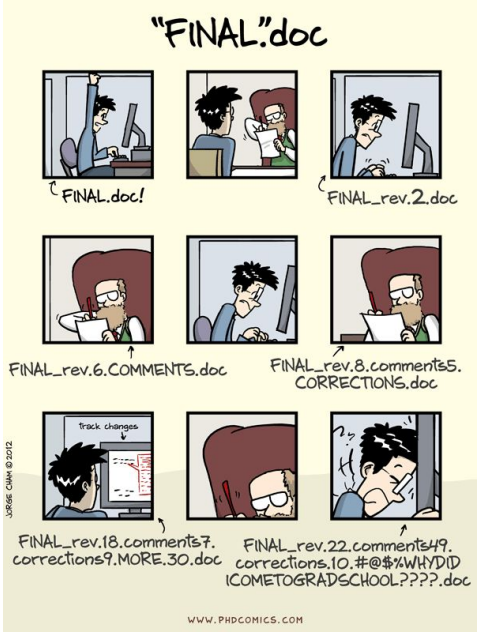
Etape 1 - Versionner le code et le partager

Pourquoi ?

- Avoir la bonne version du code
- Vision dans le temps
- Ouverture à la communauté

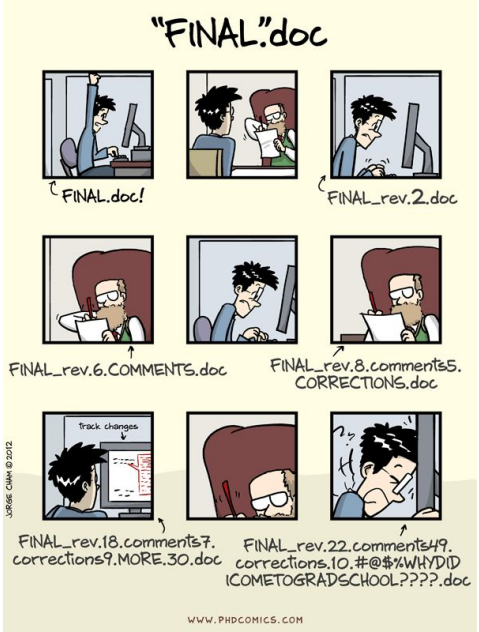
Etape 1 - Versionner le code et le partager

Avant



Etape 1 - Versionner le code et le partager

Avant



git GitHub

Avantages

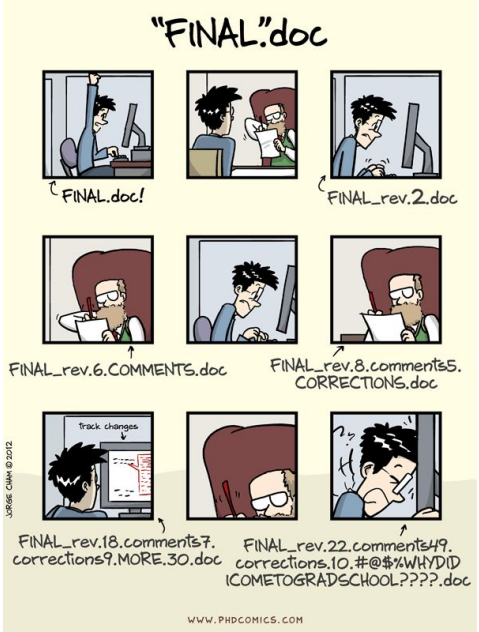
- Sauvegarde du code
- Simple pour partager
- Gestion automatique des versions

Inconvénients

- Pas simple pour les novices

Etape 1 - Versionner le code et le partager

Avant



git

GitHub

Après

thomasdenecker / FAIR_Bioinfo

Watch 3 | Star 2 | Fork 1

Code | Issues 0 | Pull requests 0 | Projects 1 | Wiki | Insights | Settings

Démonstration d'outils de bioinfo dans le cadre d'un projet

Edit

Manage topics

49 commits | 2 branches | 0 releases | 1 environment | 2 contributors | View license

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

thomasdenecker Merge branch 'master' of https://github.com/thomasdenecker/FAIR_Bioinfo — Latest commit 78990d0 10 days ago

Data	Update script	4 months ago
R-code	Change path of countable (R)	13 days ago
docs	update slide title	10 days ago
.gitignore	Add snakefile	4 months ago
Dockerfile	Improvement after first beta test	4 months ago
FAIR_app.sh	Update scripts	4 months ago
FAIR_script.sh	Remove app in FAIR_script	11 days ago
FAIR_script_session2.sh	+ accolades pour protection des variables	3 months ago
LICENCE	Create LICENCE	3 months ago
README.md	Update README.md	10 days ago
Snakefile	Update snakemake	13 days ago
_config.yml	Add dockerfile	4 months ago
annonceParCourriel.txt	2nde annonce: inscription slack	4 months ago
conditions.txt	Update scripts	4 months ago
countTable.txt	Add count table	a month ago
fair_bioinfo.qsub	File jobs	11 days ago
fair_bioinfo.slurm	update Slurm	11 days ago

Etape 2 - Création d’un environnement virtuel

Pourquoi ?

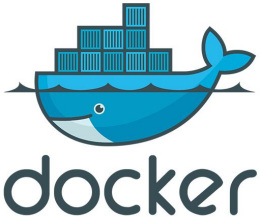
- Figer l’environnement
- Partager l’environnement

Etape 2 - Création d’un environnement virtuel

Avant

Etape 2 - Création d'un environnement virtuel

Avant



Avantages

- Rapide et léger
- Portable
- Simple à partager et déployer

Inconvénients

- Etre *root*
- Avec un système à jour

Etape 2 - Création d'un environnement virtuel

Avant

docker

Après : Ubuntu 16.04 figé

```
$ cat > dockerfile
FROM ubuntu:16.04
RUN apt-get update

# Set environment variables
ENV HOME /root

# Define working directory
WORKDIR /root

# Define default command
CMD ["bash"]

$ docker --tag=toto build .
$ docker run toto
```

Etape 3 - Installation des outils par un gestionnaire

Pourquoi ?

- Avoir la bonne version
- Installer simplement

Etape 3 - Installation des outils par un gestionnaire

Avant : exemple de FastQC

1) Télécharger la source
2) Décompresser le dossier
3) Installer et mettre à jour Java (nombreux problèmes)
4) Changer les droits

Etape 3 - Installation des outils par un gestionnaire

Avant : exemple de FastQC

1) Télécharger la source
2) Décompresser le dossier
3) Installer et mettre à jour Java (nombreux problèmes)
4) Changer les droits

Avantages

- Gestionnaire simple à installer
- Installation simple des paquets
- Gestion des versions

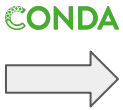
Inconvénients

- Peut être lourd (solution miniconda)
- Paquets manquants (R)

Etape 3 - Installation des outils par un gestionnaire

Avant : exemple de FastQC

1) Télécharger la source
2) Décompresser le dossier
3) Installer et mettre à jour Java (nombreux problèmes)
4) Changer les droits



Après

```
$ conda install -c bioconda -y fastqc=0.11.2
```

Tous les outils utilisés dans le protocole sont disponibles sur Conda (https://anaconda.org/) : bowtie2, samtools, htseqcount, aspera, snakemake, ...

Installation aussi simple

Etape 4 - Création d’un script d’analyse

Pourquoi ?

- Avoir un script d’analyse reproductible
- Ne pas refaire ce qui est déjà fait
- Paralléliser

Etape 4 - Création d’un script d’analyse

Avant (script Shell)

```
for sample in `ls *.fastq.gz`
do
  fastqc ${sample}
done
```

Etape 4 - Création d'un script d'analyse

Avant (script Shell)

```
for sample in `ls *.fastq.gz`
do
  fastqc ${sample}
done
```

SNAKEMAKE

Avantages

- Workflow (gestion des jobs)
- Puissant et rapide

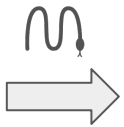
Inconvénients

- Une logique à prendre
- Syntaxe moins simple que le script shell

Etape 4 - Création d'un script d'analyse

Avant (script Shell)

```
for sample in `ls *.fastq.gz`
do
  fastqc ${sample}
done
```



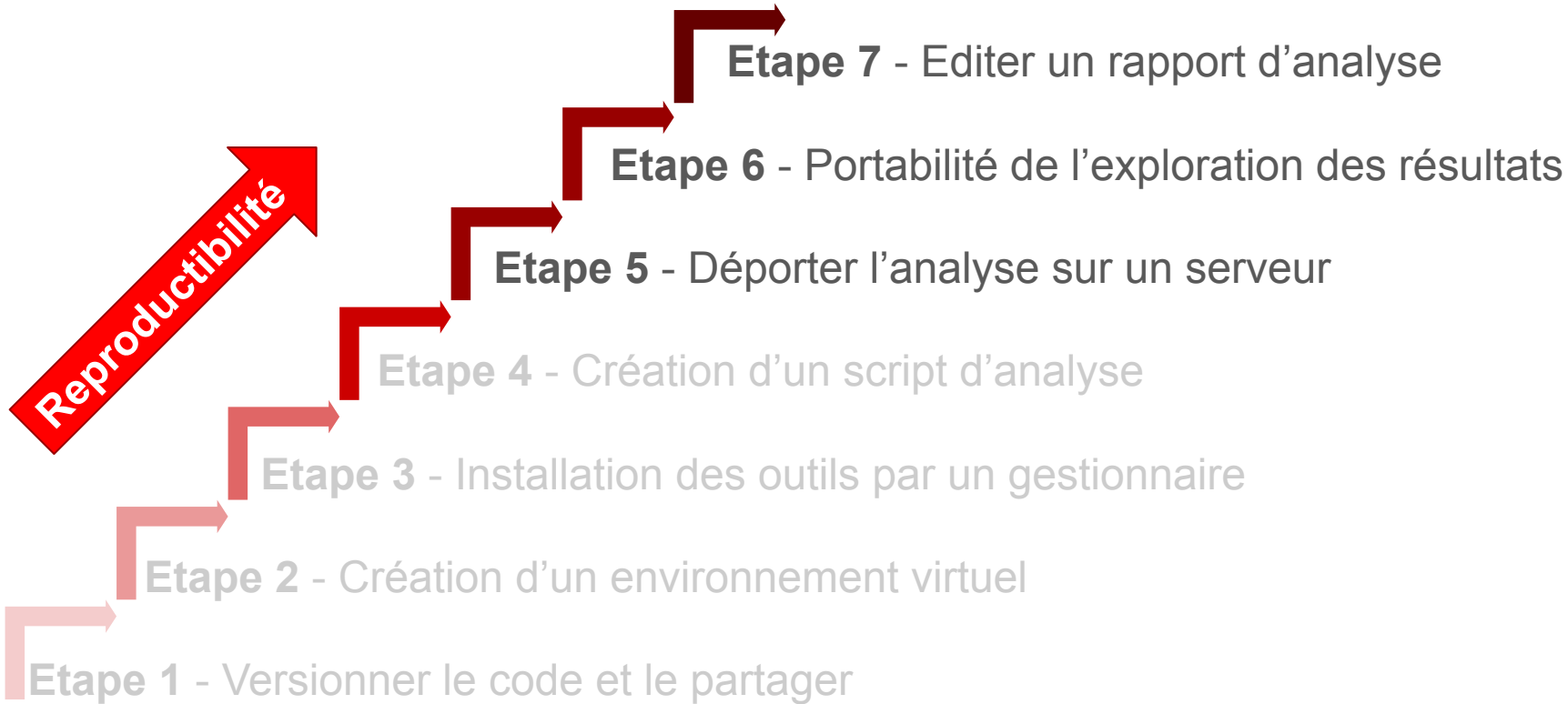
Après (Snakefile)

```
$ cat > Snakefile
SAMPLES, =
glob_wildcards("./samples/{smp}.fastq.gz")

rule final:
input:expand("fastqc/{smp}/{smp}_fastqc.zip"
,smp=SAMPLES)

rule fastqc:
   input:  "samples/{smp}.fastq.gz"
   output:"fastqc/{smp}/{smp}_fastqc.zip"
   message: """Quality check"""
   shell: """fastqc {input} --outdir
fastqc/{wildcards.smp}"""
$ snakemake
```

Plus court à écrire mais pas à exécuter

Snakemake = Parallèle

Une solution en 7 étapes



Etape 5 - Déporter l’analyse sur un serveur

Pourquoi ?

- Environnement contrôlé
- Déport de l’analyse

Etape 5 - Déporter l’analyse sur un serveur

Avant

Adaptation en local et sur les serveurs difficile voire non gérée ...

Etape 5 - Déporter l'analyse sur un serveur

Avant

Adaptation en local et sur les serveurs difficile voire non gérée ...

Avantages

- Simple à mettre en place
- Augmentation de la puissance (cloud ou cluster)
- Pour tout le monde

Inconvénients

- Pas simple pour les novices

Etape 5 - Déporter l'analyse sur un serveur

Avant

Adaptation en local et sur les serveurs difficile voire non gérée ...

Après

```
$ git clone
https://github.com/thomasdenecker/FAIR_Bioinfo

$ cd FAIR_Bioinfo

$ sudo docker run --rm -d -p 80:8888 --name
fair_bioinfo -v ${PWD}:/home/rstudio
tdenecker/fair_bioinfo bash ./FAIR_script.sh
```

Le protocole est lancé !

Etape 6 - Portabilité de l’exploration des résultats

Pourquoi ?

- Rendre simple l’exploration
- Simple à partager

Etape 6 - Portabilité de l'exploration des résultats

Avant : Terminal R

```
dds <- DESeqDataSetFromMatrix(countData =
cts,colData = coldata, design= ~ batch +
condition)

dds <- DESeq(dds)
resultsNames(dds) # lists the
coefficients
res <- results(dds, name =
"condition_trt_vs_untrt")

# or to shrink log fold changes
# association with condition:
res <- lfcShrink(dds,
coef="condition_trt_vs_untrt",
type="apeglm")
```

Etape 6 - Portabilité de l'exploration des résultats

Avant : Terminal R

```
dds <- DESeqDataSetFromMatrix(countData =
cts,colData = coldata, design= ~ batch +
condition)

dds <- DESeq(dds)
resultsNames(dds) # lists the
coefficients
res <- results(dds, name =
"condition_trt_vs_untrt")

# or to shrink log fold changes
# association with condition:
res <- lfcShrink(dds,
coef="condition_trt_vs_untrt",
type="apeglm")
```

Avantages

- Portable (HTML)
- Accessible partout
- Interactif (paramétrable, graphes dynamiques, ...)

Inconvénients

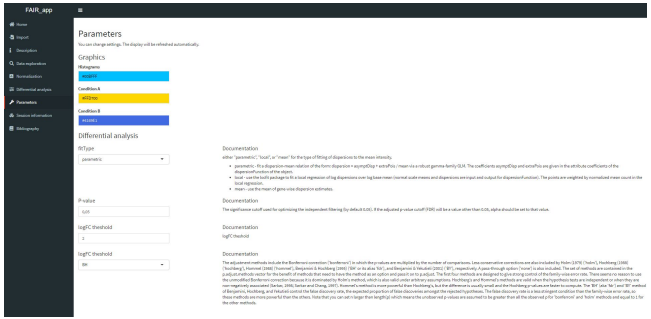
- Mélange de R et de HTML

Etape 6 - Portabilité de l'exploration des résultats

Avant : Terminal R

```
dds <- DESeqDataSetFromMatrix(countData =
cts,colData = coldata, design= ~ batch +
condition)

dds <- DESeq(dds)
resultsNames(dds) # lists the
coefficients
res <- results(dds, name =
"condition_trt_vs_untrt")

# or to shrink log fold changes
# association with condition:
res <- lfcShrink(dds,
coef="condition_trt_vs_untrt",
type="apeglm")
```

Après



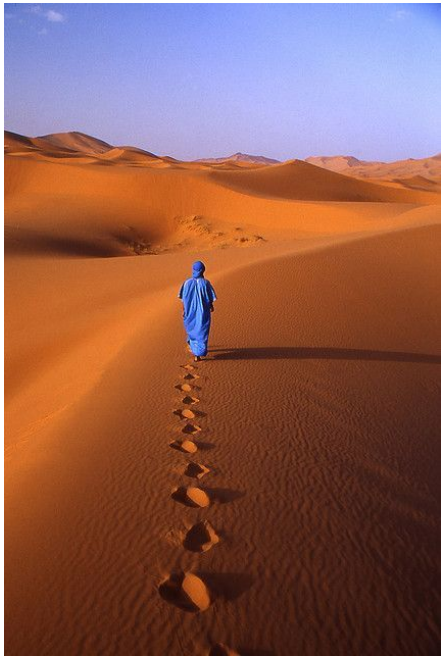
Etape 7 - Editer un rapport d’analyse

Pourquoi ?

- Avoir une trace de l’analyse
  (date, heure, paramètres, …)
- Stocker les versions des outils

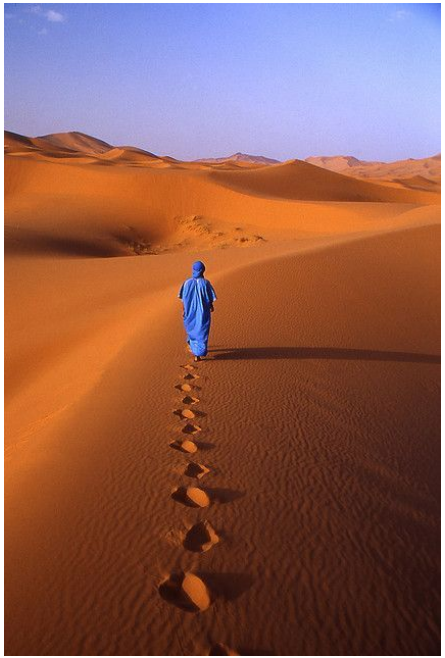
Etape 7 - Editer un rapport d’analyse

Avant



Etape 7 - Editer un rapport d'analyse

Avant



Avantages

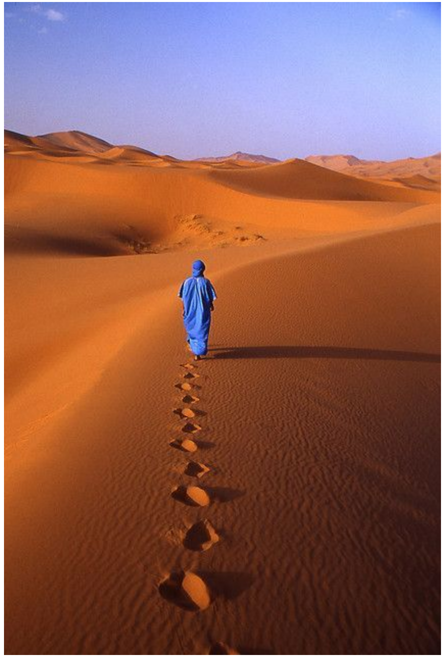
- Syntaxe simple (Markdown)
- Partage (PDF, HTML, ...)

Inconvénients

- Rares problèmes de visualisation en LaTeX

Etape 7 - Editer un rapport d’analyse

Avant



rmarkdown

jupyter

Après

Rapport

Description of raw data

The objective of this application is to find the differentially expressed genes after using the FAIR_Bioinfo workflow

Conditions

The count data files and associated biological conditions are listed in the following table :

```
##     SampleID     Iron Light Time Condition_Name
## 1 SRR3105699 DEPLETED LIGHT   9H          CondA
## 2 SRR3105698 DEPLETED LIGHT   9H          CondA
## 3 SRR3105697 DEPLETED LIGHT   9H          CondA
## 4 SRR3099587 STANDARD LIGHT   9H          CondB
## 5 SRR3099586 STANDARD LIGHT   9H          CondB
## 6 SRR3099585 STANDARD LIGHT   9H          CondB
##                                                                    wget
## 1 ftp.sra.ebi.ac.uk/vol1/fastq/SRR310/009/SRR3105699/SRR3105699.fastq.gz
## 2 ftp.sra.ebi.ac.uk/vol1/fastq/SRR310/008/SRR3105698/SRR3105698.fastq.gz
## 3 ftp.sra.ebi.ac.uk/vol1/fastq/SRR310/007/SRR3105697/SRR3105697.fastq.gz
## 4 ftp.sra.ebi.ac.uk/vol1/fastq/SRR309/007/SRR3099587/SRR3099587.fastq.gz
## 5 ftp.sra.ebi.ac.uk/vol1/fastq/SRR309/006/SRR3099586/SRR3099586.fastq.gz
## 6 ftp.sra.ebi.ac.uk/vol1/fastq/SRR309/005/SRR3099585/SRR3099585.fastq.gz
##                                                                                 ascp
## 1 era-fasp@fasp.sra.ebi.ac.uk:/vol1/fastq/SRR310/009/SRR3105699/SRR3105699.fastq.gz
## 2 era-fasp@fasp.sra.ebi.ac.uk:/vol1/fastq/SRR310/008/SRR3105698/SRR3105698.fastq.gz
## 3 era-fasp@fasp.sra.ebi.ac.uk:/vol1/fastq/SRR310/007/SRR3105697/SRR3105697.fastq.gz
## 4 era-fasp@fasp.sra.ebi.ac.uk:/vol1/fastq/SRR309/007/SRR3099587/SRR3099587.fastq.gz
## 5 era-fasp@fasp.sra.ebi.ac.uk:/vol1/fastq/SRR309/006/SRR3099586/SRR3099586.fastq.gz
## 6 era-fasp@fasp.sra.ebi.ac.uk:/vol1/fastq/SRR309/005/SRR3099585/SRR3099585.fastq.gz
```

Table 1: Data files and associated biological conditions.

Count table

```
##                 CondA_SRR3105699 CondA_SRR3105698 CondA_SRR3105697
## ostta01g00010              215              403              199
## ostta01g00020              166              339              195
## ostta01g00030               51               75               55
## ostta01g00040              671             1444              847
## ostta01g00050              368              970              354
## ostta01g00060              197              363              202
##                 CondB_SRR3099587 CondB_SRR3099586 CondB_SRR3099585
## ostta01g00010              395              415              305
## ostta01g00020              252              287              156
## ostta01g00030               57               29               23
## ostta01g00040              946              984              725
## ostta01g00050              558              592              500
## ostta01g00060              299              249              226
```

Table 2: View of the count data table.

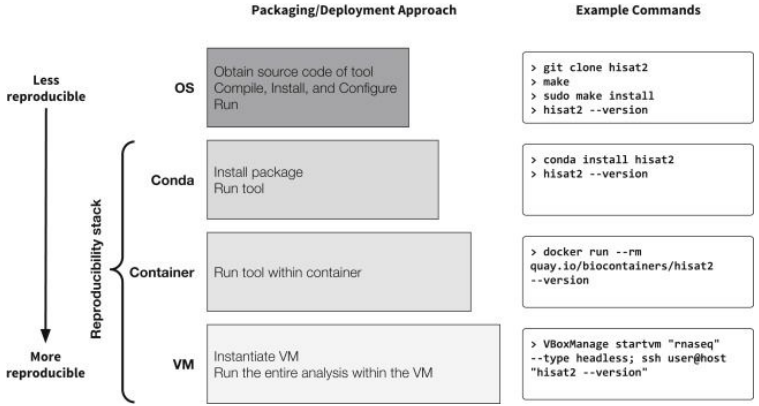
Looking at the summary of the count table provides a basic description of these raw counts (min and max values, median, etc).

```
##  CondA_SRR3105699  CondA_SRR3105698  CondA_SRR3105697  CondB_SRR3099587
##  Min.   :    0.0   Min.   :    0.0   Min.   :    0.0   Min.   :    0.0
##  1st Qu.:   60.0   1st Qu.:  122.0   1st Qu.:   68.0   1st Qu.:   85.0
##  Median :  147.0   Median :  313.0   Median :  162.0   Median :  208.0
##  Mean   :  353.3   Mean   :  797.4   Mean   :  402.4   Mean   :  531.8
##  3rd Qu.:  364.0   3rd Qu.:  791.0   3rd Qu.:  394.0   3rd Qu.:  505.0
##  Max.   :21362.0   Max.   :58521.0   Max.   :21909.0   Max.   :41471.0
##  CondB_SRR3099586  CondB_SRR3099585
##  Min.   :    0.0   Min.   :    0.0
##  1st Qu.:   72.0   1st Qu.:   40.0
##  Median :  177.0   Median :  127.0
##  Mean   :  471.8   Mean   :  467.0
##  3rd Qu.:  433.0   3rd Qu.:  370.5
##  Max.   :41359.0   Max.   :42448.0
```

Table 3: Summary of the raw counts.

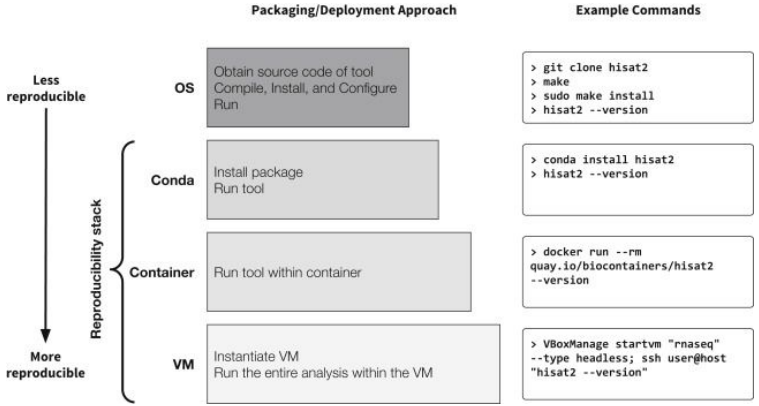
Conclusion

Quel est notre niveau de reproductibilité?



Practical Computational Reproducibility in the Life Sciences, Björn Grüning *et al*, 2018

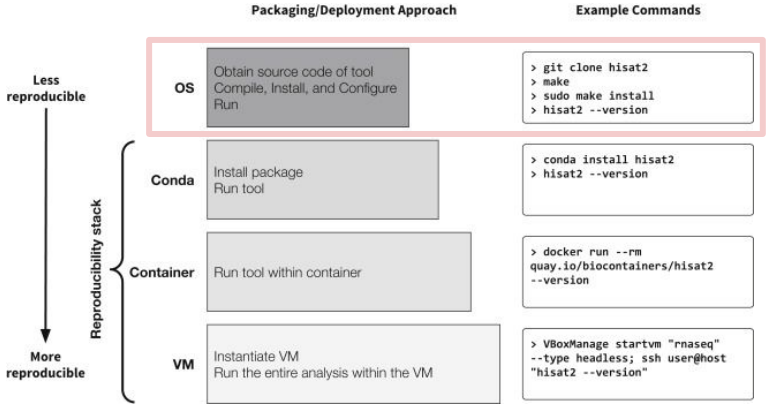
Quel est notre niveau de reproductibilité?



Practical Computational Reproducibility in the Life Sciences, Björn Grüning *et al*, 2018

FAIR bioinfo

Quel est notre niveau de reproductibilité?

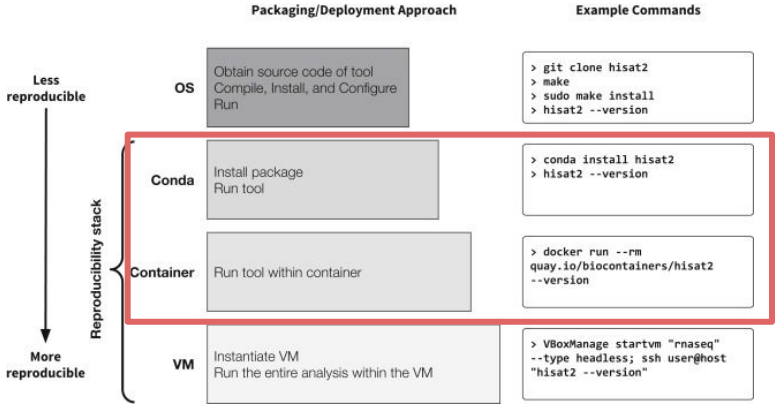


Practical Computational Reproducibility in the Life Sciences, Björn Grüning *et al*, 2018

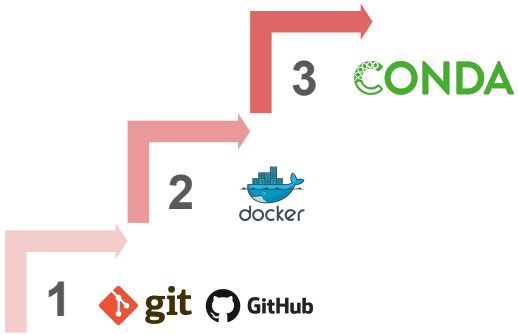
1 git GitHub

FAIR bioinfo

Quel est notre niveau de reproductibilité?

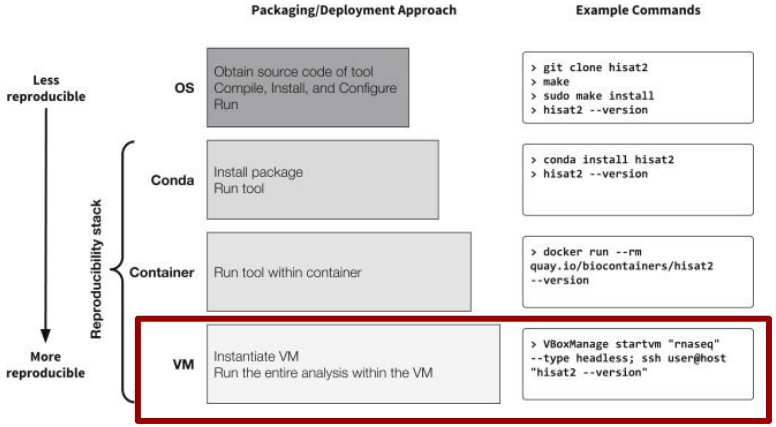


Practical Computational Reproducibility in the Life Sciences, Björn Grüning *et al*, 2018

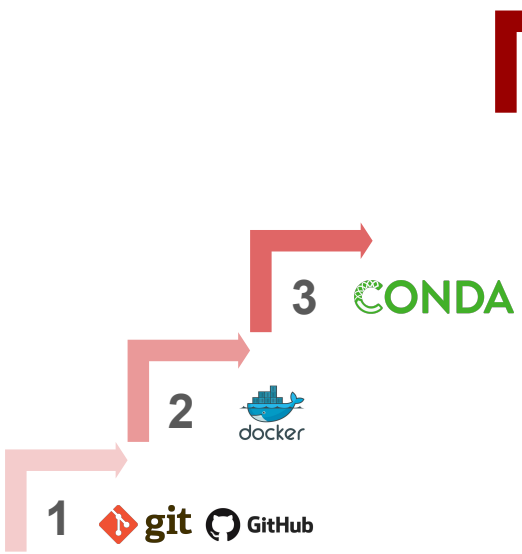


FAIR bioinfo

Quel est notre niveau de reproductibilité?

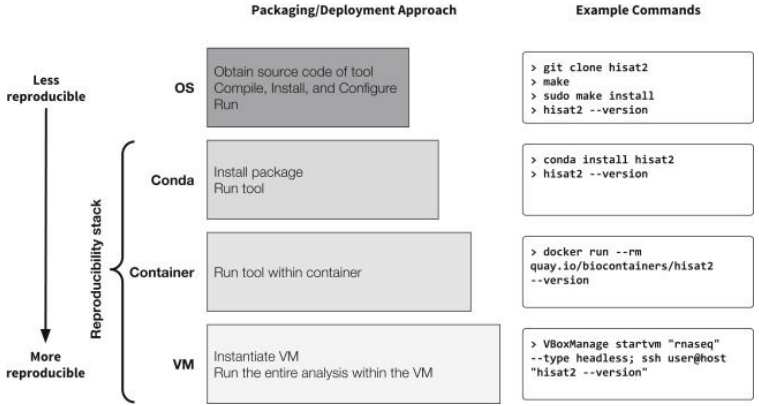


Practical Computational Reproducibility in the Life Sciences, Björn Grüning *et al*, 2018

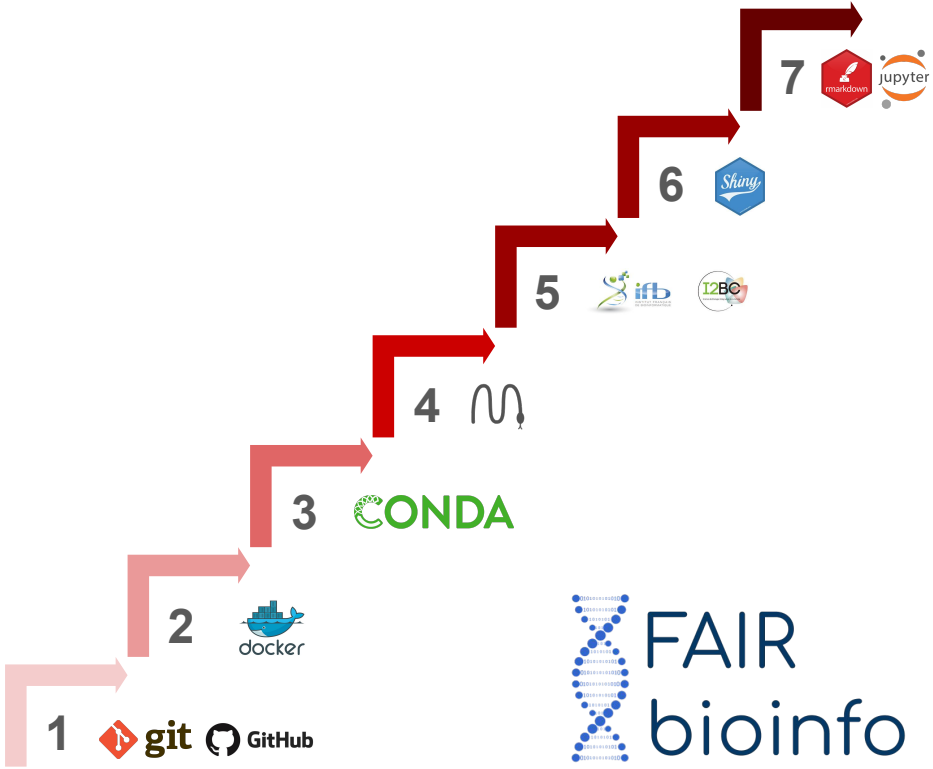


FAIR bioinfo

Quel est notre niveau de reproductibilité?



Practical Computational Reproducibility in the Life Sciences, Björn Grüning *et al*, 2018



FAIR bioinfo

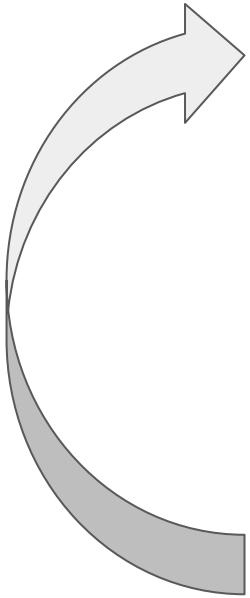
Take home messages

Une vraie réflexion sur la reproductibilité des analyses en Bioinformatique

Proposition d'une solution qui aide à rendre reproductible n'importe quel protocole d'analyse

La reproductibilité est une plus value pour la Bioinformatique !

Un cercle vertueux



FAIR raw data

+

FAIR_bioinfo scripts/protocols

=

FAIR processed data