

Pourquoi et comment créer du code “FAIR” ?

Claire TOFFANO-NIOCHE

6 juin 2019

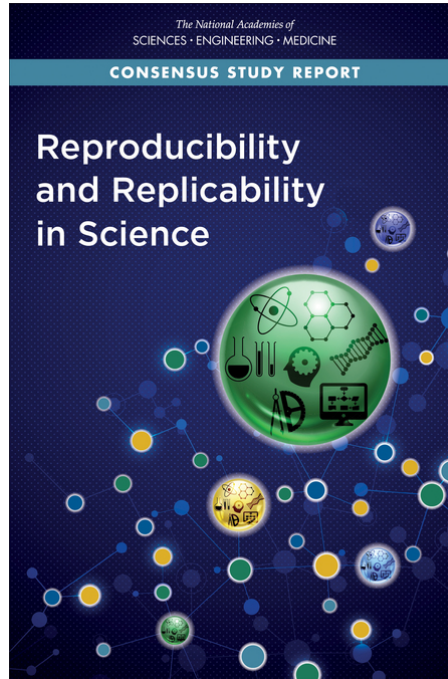
PEPI IBIS (Ingénierie Bio Informatique et Statistique)

Session thématique "Reproductibilité - Containerisation - Nextflow / Snakemake"



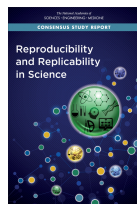
Thomas DENECKER

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>.

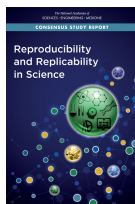
Qu'est-ce que la reproductibilité ?



Reproductibilité

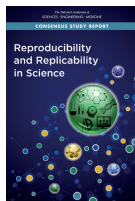
Obtenir des résultats de calcul cohérents en utilisant les mêmes données d'entrée, étapes de calcul, méthodes, codes et conditions d'analyses.

La reproductibilité *versus* la répliquabilité



Reproductibilité

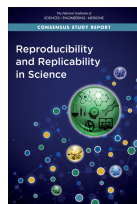
Obtenir des résultats de calcul cohérents en utilisant les mêmes données d'entrée, étapes de calcul, méthodes, codes et conditions d'analyses.



Répliquabilité

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

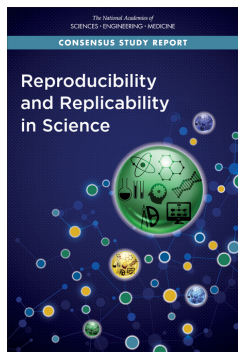
Obtenir des résultats de calcul cohérents en utilisant les mêmes données d'entrée, étapes de calcul, méthodes, codes et conditions d'analyses.



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)

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 ?

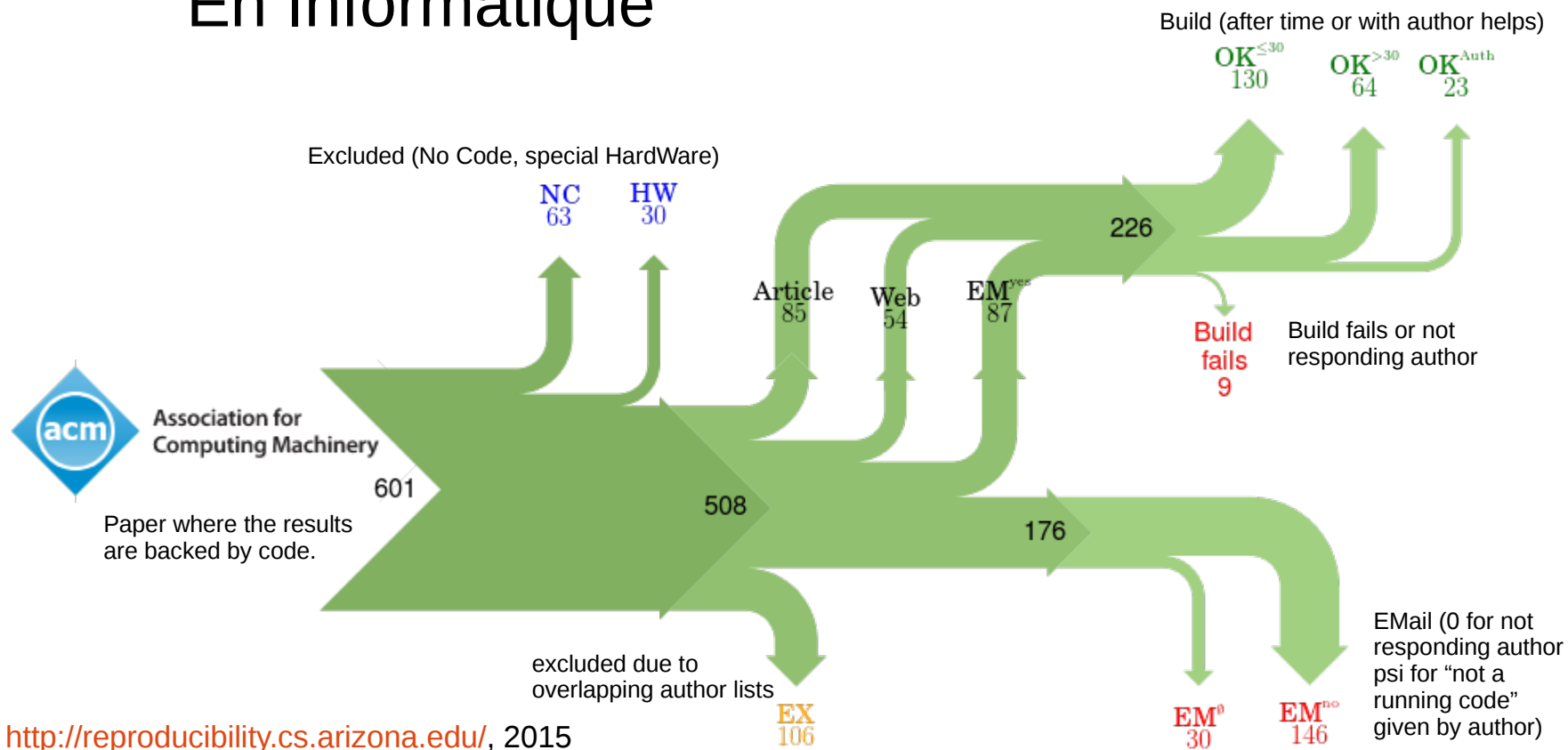
Un problème de reproductibilité en Biologie

70 %

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

(Monya Baker, *Nature*, 2016)

En Informatique



<http://reproducibility.cs.arizona.edu/>, 2015

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 inutilisables 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



...

Un exemple de reproductibilité en Bioinformatique

Formation de nos collègues à l'I2BC

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

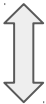
Tous les cours et les codes sont *open source*

https://github.com/thomasdenecker/FAIR_Bioinfo



Proposition d'une solution

Des équivalents pour chaque choix



...



Les principes FAIR data

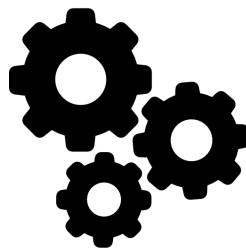
Findable



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)



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

+

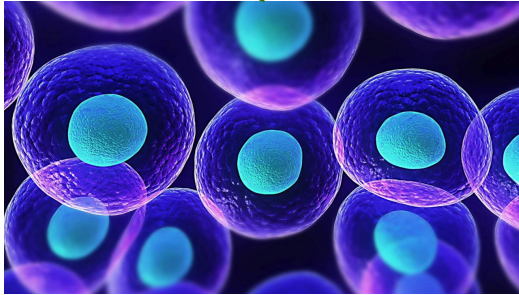
FAIR_bioinfo scripts/protocols

=

FAIR processed data

Les données : un peu de biologie ?

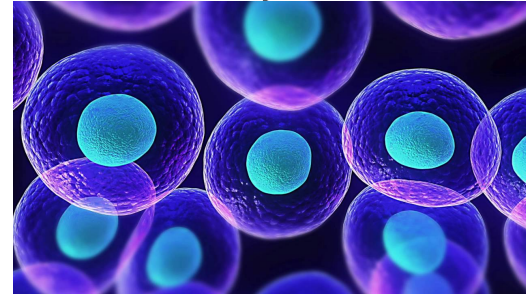
Stress



Condition 1 (3x)

Mesure 1 (3x)

No stress



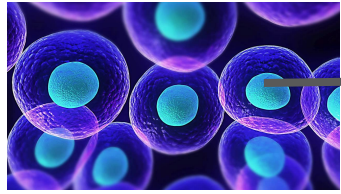
Condition 2 (3x)

Mesure 2 (3x)

Répétabilité

Différences ?

Niveau d'expression des gènes



Cellules dans une condition

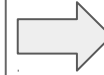
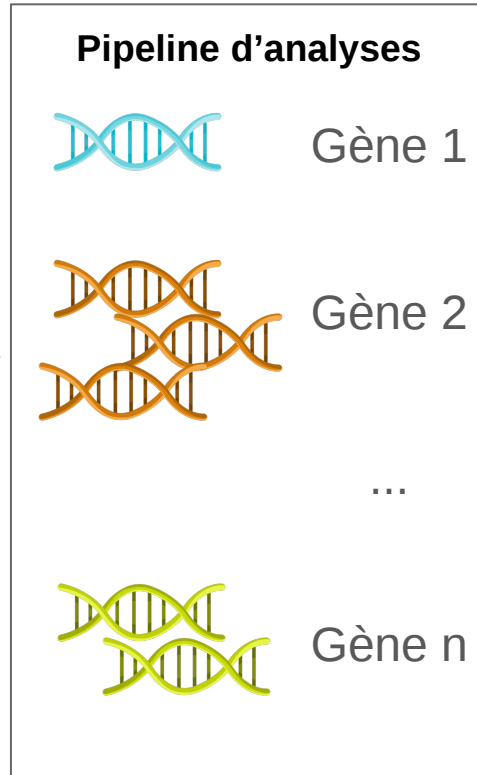
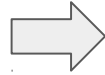
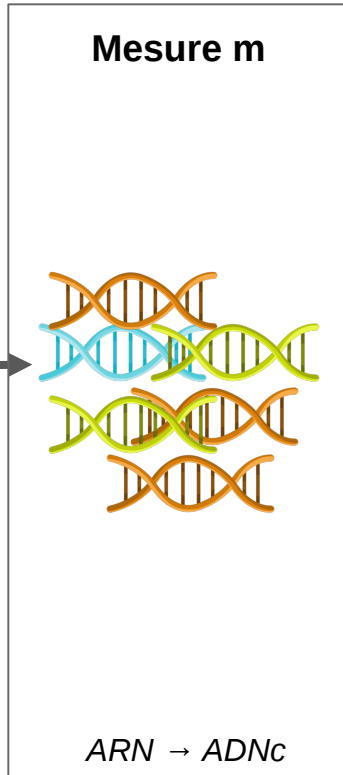


Table de comptage

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

≠

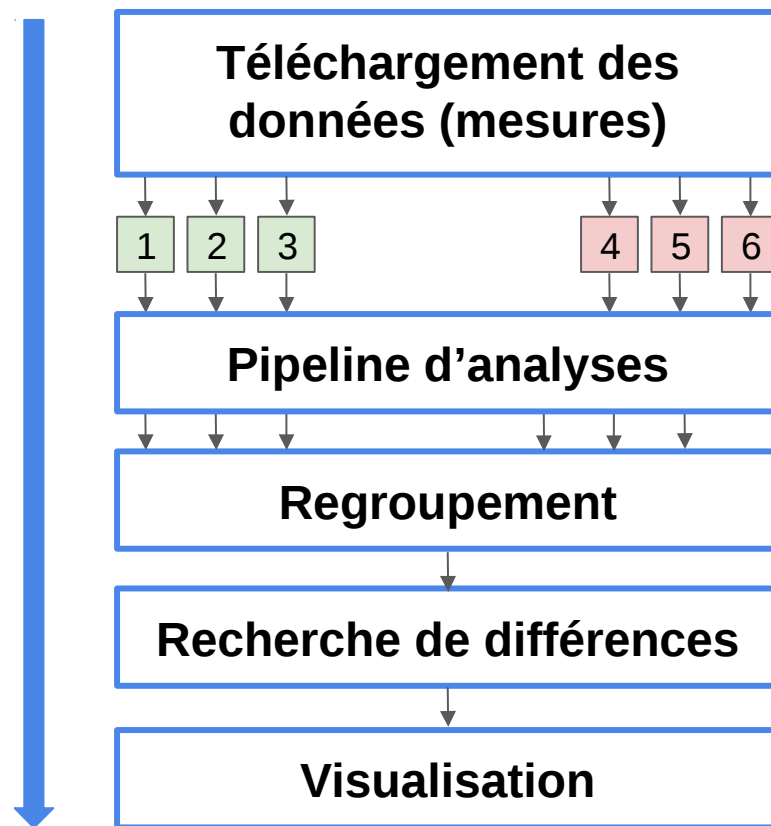
≠

=

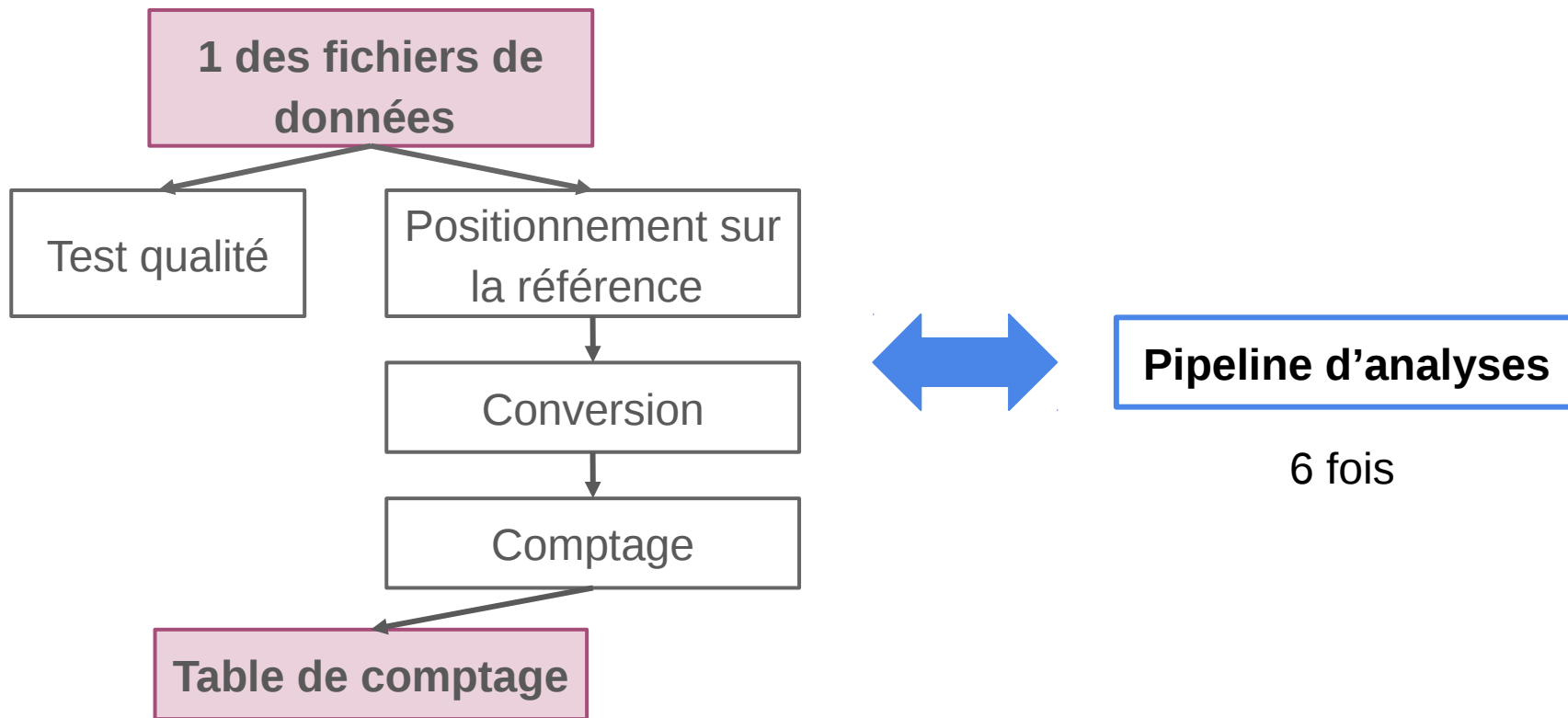
Conclusion

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

Comment traiter ces données ?



Pipeline d'analyses



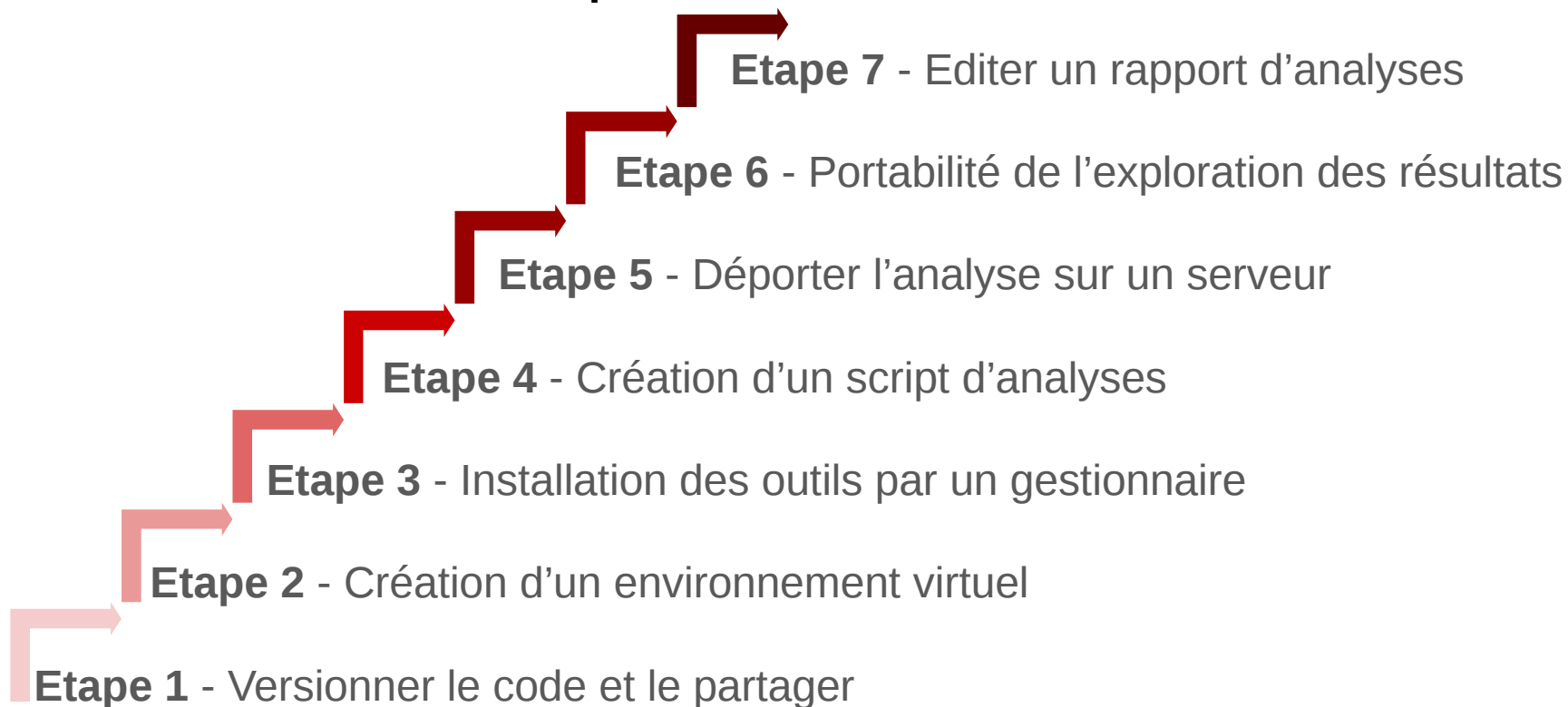
Ce qui se fait majoritairement

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, ...)

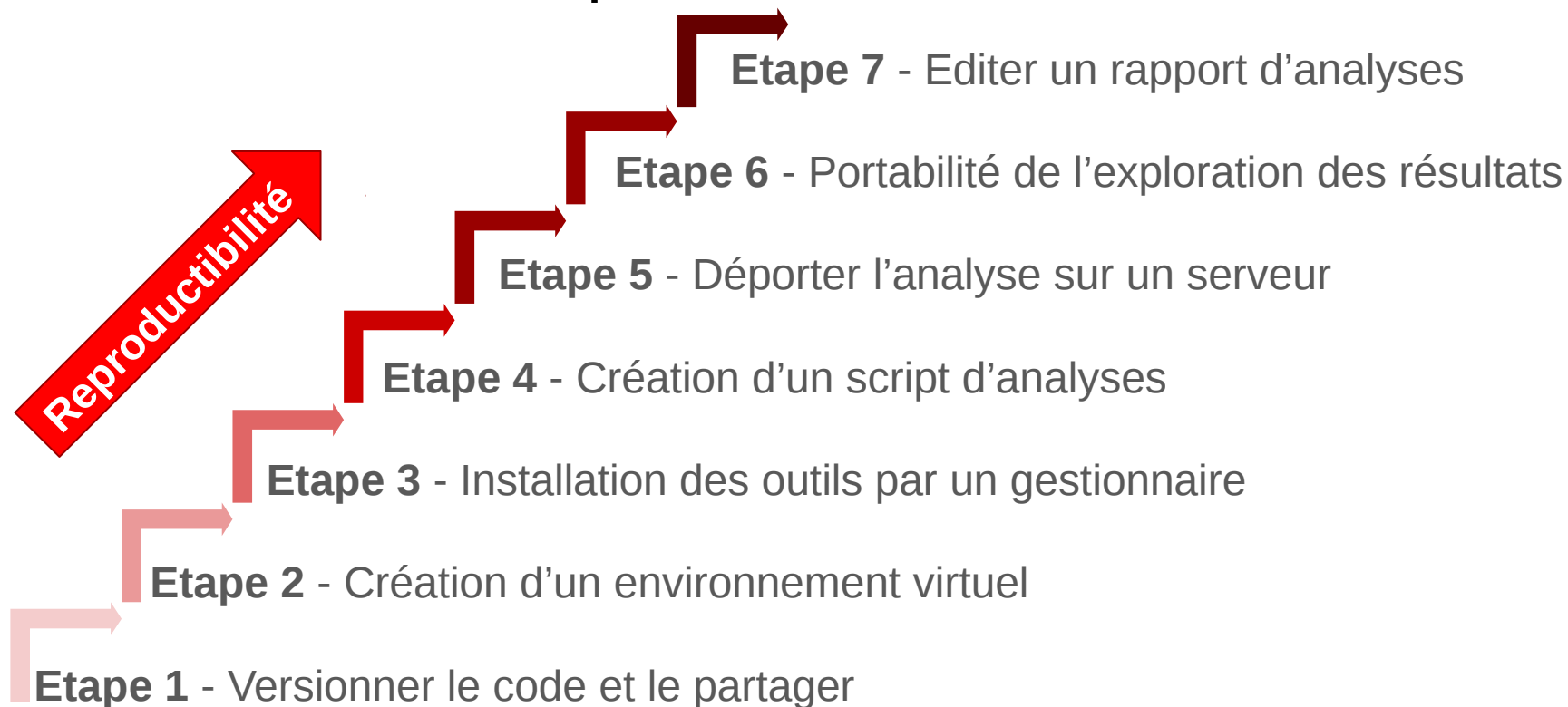
Mais peu de reproductibilité ...

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

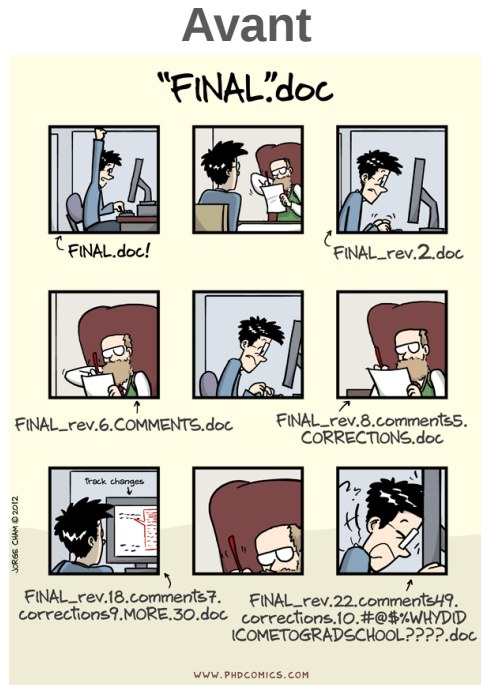


Etape 1 - Versionner le code et le partager

Avant



Etape 1 - Versionner le code et le partager



git



GitHub

Avantages

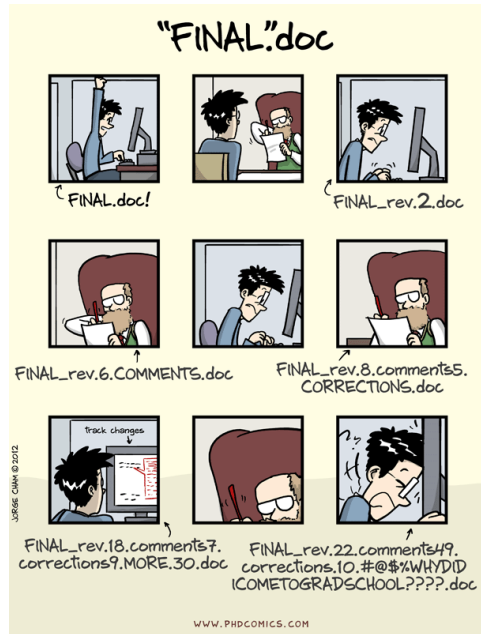
- Sauvegarde du code
- Simple pour partager
- Gestion automatique des versions

Inconvénients

- Pas simple pour les novices

Etape 1 - Versionner le code et le partager

Avant



Après

thomasdeneker / FAIR_Bioinfo

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

Branch: master New pull request

File	Description	Last commit
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 ?

- Figurer l'environnement
- Partager l'environnement



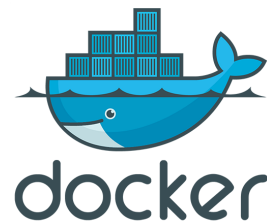
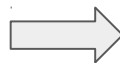
Etape 2 - Création d'un environnement virtuel

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



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

CONDA



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

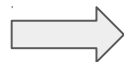
Avant (script Shell)

```
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
```



Avantages

- Workflow (gestion des jobs)
- Puissant et rapide

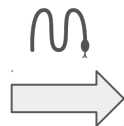
Inconvénients

- 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
```



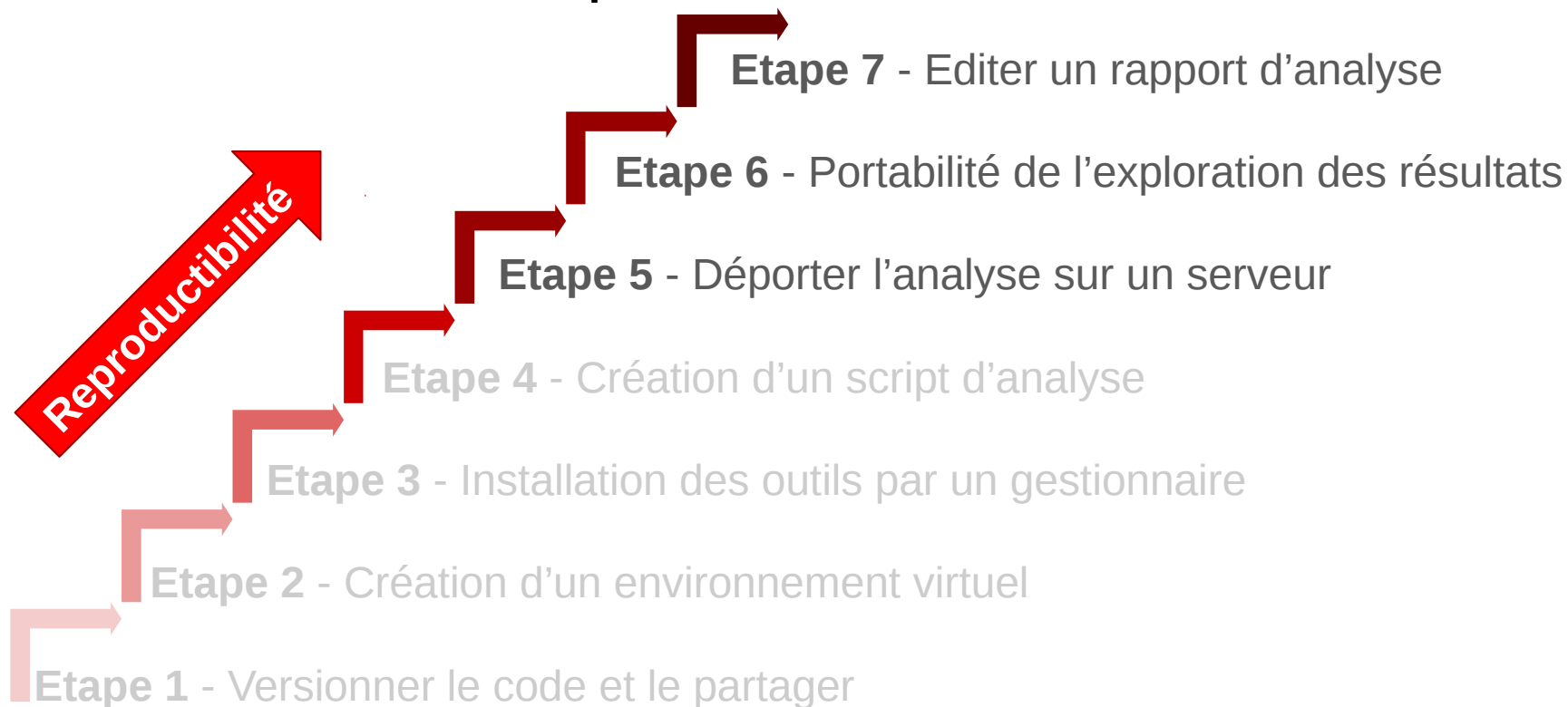
Après (Snakefile)

```
$ 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")
```

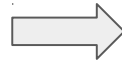
Etape 6 - Portabilité de l'exploration des résultats

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

The screenshot shows the FAIR_app Shiny interface. The top panel is the 'Parameters' tab, which includes sections for 'Graphics' (with 'Background' set to 'white'), 'Differential analysis' (with 'betaType' set to 'parameters'), and 'P-value' (with 'log' selected). The bottom panel is the 'Differential analysis' tab, which includes sections for 'Modification', 'Outlier detection', and 'Dispersion estimation'. The 'Dispersion estimation' section contains a plot of 'log2 fold change' vs 'base of normalized counts' and a histogram of 'log-normality dispersion estimate'. The plot shows a distribution of log2 fold changes centered around zero, with a red line indicating the expected distribution. The histogram shows the distribution of log-normality dispersion estimates, which is centered around 1.0.

Etape 7 - Editer un rapport d'analyse

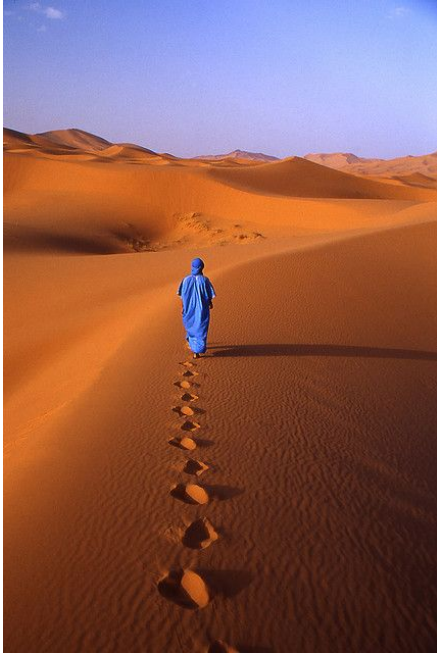
Pourquoi ?

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



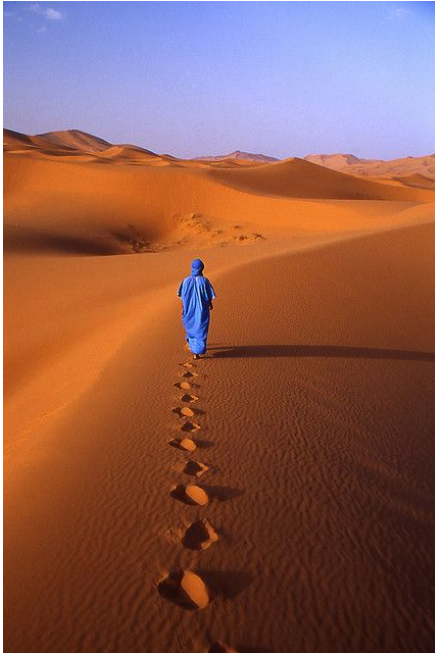
Etape 7 - Editer un rapport d'analyse

Avant



Etape 7 - Editer un rapport d'analyse

Avant



Avantages

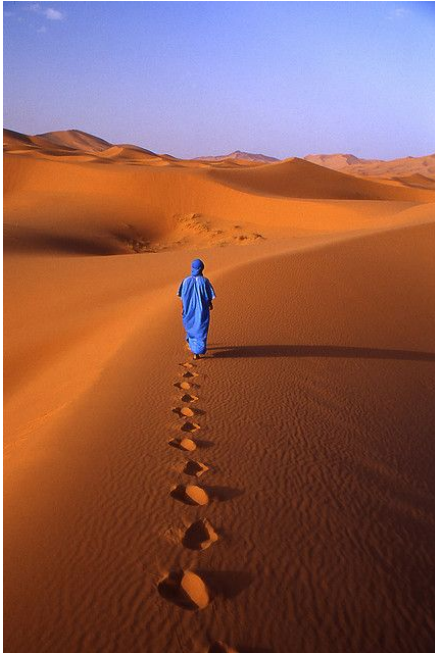
- Syntaxe simple (Markdown)
- Partage (PDF, HTML, ...)

Inconvénients

- Rares problèmes de visualisation en $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$

Etape 7 - Editer un rapport d'analyse

Avant



Après

Rapport

Description of raw data

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

Conditions

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

```
## SampleID Iron Light Time Condition_Name
## 1 SR130509 DEPLETED 150T 9h Conda
## 2 SR130508 DEPLETED 150T 9h Conda
## 3 SR130507 DEPLETED 150T 9h Conda
## 4 SR309587 STANDARD 150T 9h Conda
## 5 SR309586 STANDARD 150T 9h Conda
## 6 SR309585 STANDARD 150T 9h Conda
##
## 7 fig.vra.ebi.ac.uk/vol1/Factory/SR130/008/SR130509/SR130509_Factory
## 8 fig.vra.ebi.ac.uk/vol1/Factory/SR130/008/SR130508/SR130508_Factory
## 9 fig.vra.ebi.ac.uk/vol1/Factory/SR130/007/SR130507/SR130507_Factory
## 4 fig.vra.ebi.ac.uk/vol1/Factory/SR309/008/SR309587/SR309587_Factory
## 5 fig.vra.ebi.ac.uk/vol1/Factory/SR309/008/SR309586/SR309586_Factory
## 6 fig.vra.ebi.ac.uk/vol1/Factory/SR309/008/SR309585/SR309585_Factory
##
## 3 era-FacqFaq.vra.ebi.ac.uk/vol1/Factory/SR130/008/SR130509/SR130509_Factory
## 2 era-FacqFaq.vra.ebi.ac.uk/vol1/Factory/SR130/008/SR130508/SR130508_Factory
## 3 era-FacqFaq.vra.ebi.ac.uk/vol1/Factory/SR130/007/SR130507/SR130507_Factory
## 4 era-FacqFaq.vra.ebi.ac.uk/vol1/Factory/SR309/007/SR309587/SR309587_Factory
## 5 era-FacqFaq.vra.ebi.ac.uk/vol1/Factory/SR309/008/SR309586/SR309586_Factory
## 6 era-FacqFaq.vra.ebi.ac.uk/vol1/Factory/SR309/008/SR309585/SR309585_Factory
```

Table 1: Data files and associated biological conditions

Count table

```
## Conda_SR130509 Conda_SR130508 Conda_SR130507
## outtag000a 255 403 199
## outtag000b 166 339 195
## outtag000c 51 75 55
## outtag000d 671 1444 847
## outtag000e 368 976 254
## outtag000f 197 363 262
## Conda_SR309587 Conda_SR309586 Conda_SR309585
## outtag000a 395 435 195
## outtag000b 252 287 156
## outtag000c 57 29 23
## outtag000d 966 984 725
## outtag000e 558 592 588
## outtag000f 299 249 226
```

Table 2: View of the count data table

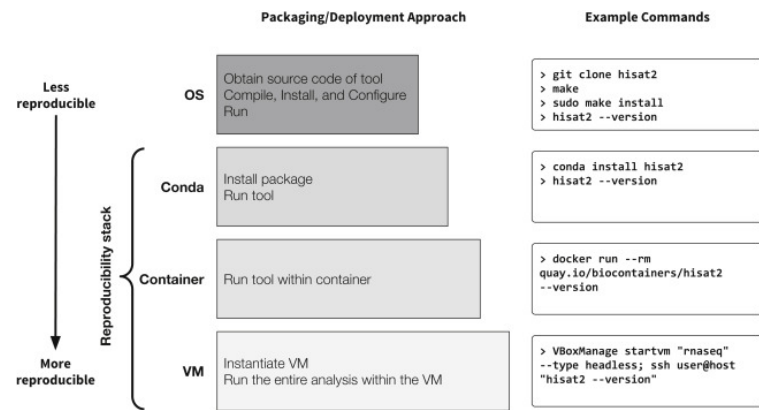
Looking at the summary of the count table provides a basic description of these raw counts (min and max values, median, etc).

```
## Conda_SR130509 Conda_SR130508 Conda_SR130507 Conda_SR309587
## Min.: 0.0 Min.: 0.0 Min.: 0.0 Min.: 0.0
## 1st Qu.: 69.0 1st Qu.: 122.0 1st Qu.: 68.0 1st Qu.: 85.0
## Median: 187.0 Median: 333.0 Median: 102.0 Median: 208.0
## Mean: 351.3 Mean: 792.0 Mean: 462.4 Mean: 521.8
## 3rd Qu.: 364.0 3rd Qu.: 799.0 3rd Qu.: 364.0 3rd Qu.: 507.0
## Max.: 22302.0 Max.: 158222.0 Max.: 22809.0 Max.: 142473.0
## Conda_SR309586 Conda_SR309585
## Min.: 0.0 Min.: 0.0
## 1st Qu.: 72.0 1st Qu.: 49.0
## Median: 377.0 Median: 137.0
## Mean: 471.8 Mean: 467.0
## 3rd Qu.: 413.0 3rd Qu.: 376.5
## Max.: 14195.0 Max.: 142468.0
```

Table 3: Summary of the raw counts

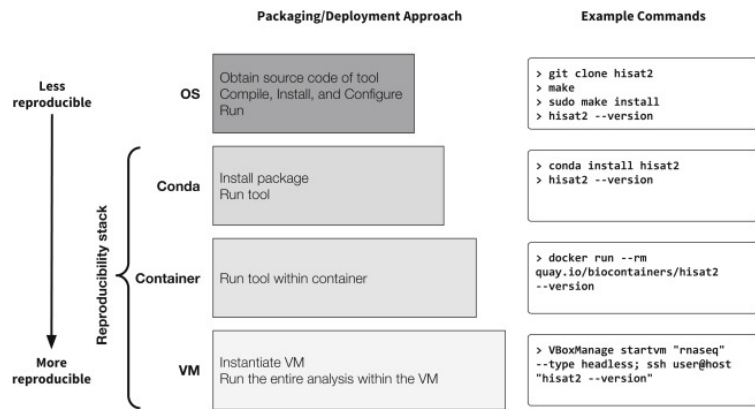
Conclusion

Quel est notre niveau de reproductibilité?



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

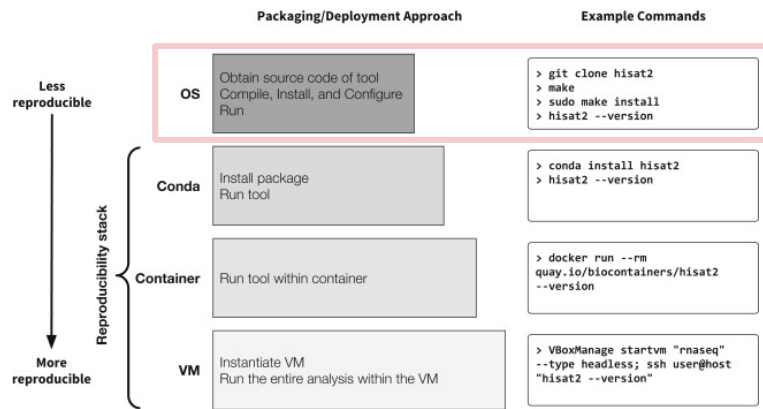
Quel est notre niveau de reproductibilité?



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



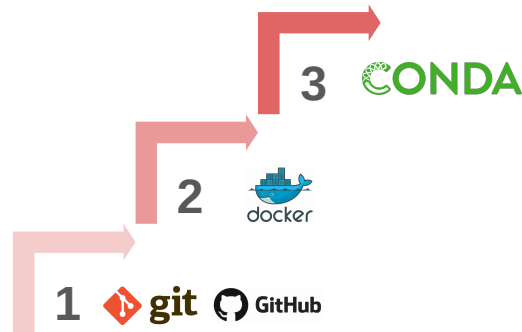
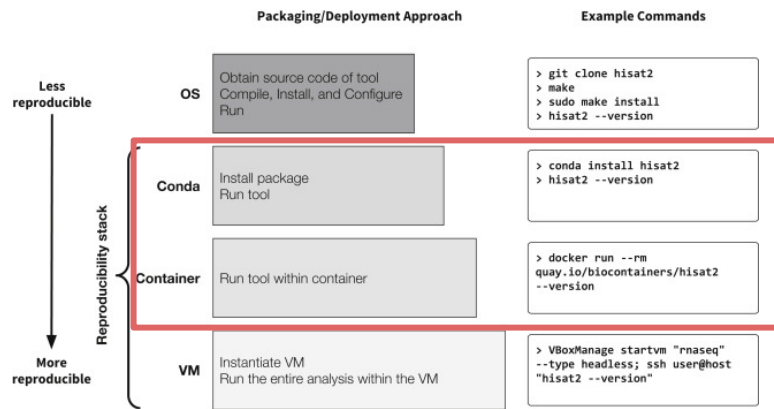
Quel est notre niveau de reproductibilité?



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



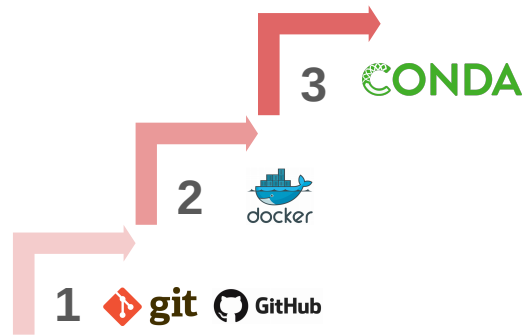
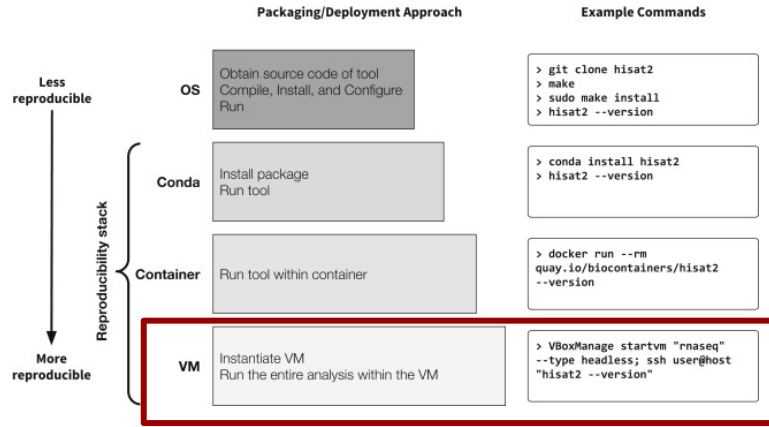
Quel est notre niveau de reproductibilité?



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



Quel est notre niveau de reproductibilité?

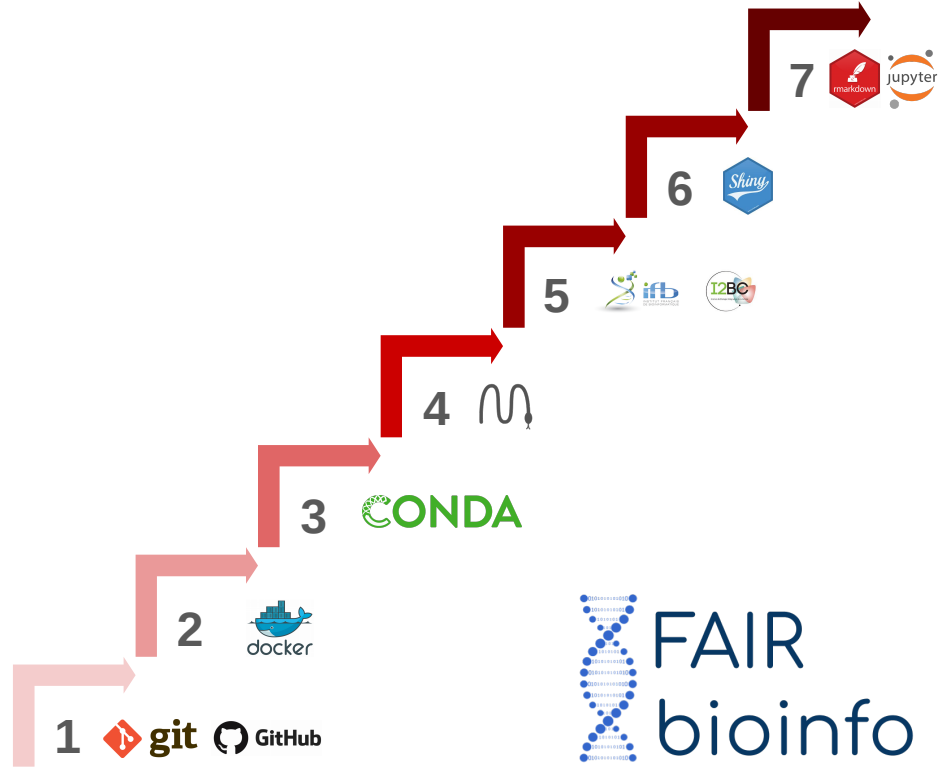


Practical Computational Reproducibility
 in the Life Sciences, Björn Grüning *et al*,
 2018



Quel est notre niveau de reproductibilité?

	Packaging/Deployment Approach	Example Commands
Less reproducible ↓ More reproducible Reproducibility stack	OS Obtain source code of tool Compile, Install, and Configure Run	<pre>> git clone hisat2 > make > sudo make install > hisat2 --version</pre>
	Conda Install package Run tool	<pre>> conda install hisat2 > hisat2 --version</pre>
	Container Run tool within container	<pre>> docker run --rm quay.io/biocontainers/hisat2 --version</pre>
	VM Instantiate VM Run the entire analysis within the VM	<pre>> VBoxManage startvm "rnaseq" --type headless; ssh user@host "hisat2 --version"</pre>



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



Limites

FAIR_bioinfo :

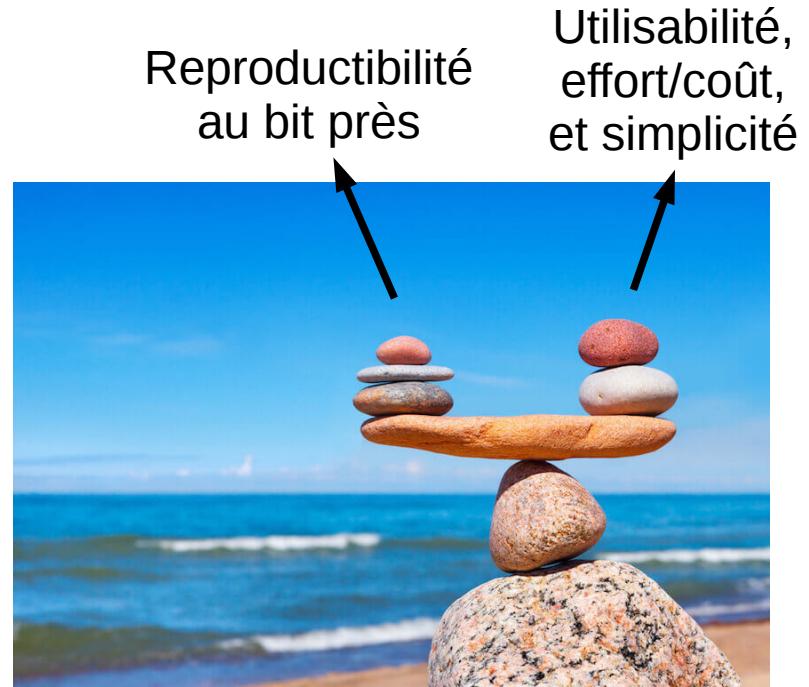
- usage d'une VM déjà instanciée
- mieux = créer sa propre image de VM

Reproductibilité bit à bit ?

- conteneur exploite qq unes des ressources de la machine support
- mieux = contrôle de version de l'environnement (Nix, Guix)

Parallélisation :

- perte de l'ordre des calculs, multi-threading, même matériel ?
- mieux = ...?



Take home messages

Une vraie réflexion sur la reproductibilité des analyses en Bioinformatique

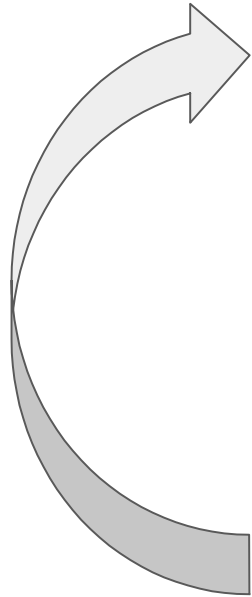
Proposition d'une solution qui aide à rendre reproductible n'importe quel protocole d'analyse

La reproductibilité est une plus value pour la Bioinformatique !

Pour retrouver une vidéo de cette présentation :

<https://webcast.in2p3.fr/video/la-reproductibilite-au-service-de-la-biologie-computationnelle>

Un cercle vertueux



FAIR raw data

+

FAIR_bioinfo scripts/protocols

=

FAIR processed data