Bonnes pratiques en bioinformatique : (essayer) d'aller vers plus de reproductibilité

DUBii - Module 5

Valentin Loux - Cédric Midoux - Olivier Rué

2020/06/11

Programme de l'après-midi

- Intro : Reproductibilité, science ouverte
- Partie 1 : Organiser son espace de travail
- Partie 2 : Des langages à faible balisage pour faciliter la traçabilité et la prise de note
- Partie 3 : Versionner ses documents
  - TP
- Partie 4 : Utilisation de documents computationels -- Notebook
  - TP
- Conclusion et pour aller + loin

Tout le monde a déjà eu cette experience

OPEN ACCESS Freely available online PLoS one

The *Arthrobacter arilaitensis* Re117 Genome Sequence Reveals Its Genetic Adaptation to the Surface of Cheese

Christophe Monnet[1,2]*, Valentin Loux[3], Jean-François Gibrat[3], Eric Spinnler[1,2], Valérie Barbe[4], Benoit Vacherie[4], Frederick Gavory[4], Edith Gourbeyre[5], Patricia Siguier[5], Michaël Chandler[5], Rayda Elleuch[6], Françoise Irlinger[1,2☯], Tatiana Vallaeys[7☯]

Un Papier interessant

collaboration with the user community. Genome comparisons were performed using Origami, an in-house tool developed for microbial genome comparison. Orthologs were defined as reciprocal best hits with an e-value lower than 10^{-3}. Transposases were excluded from the analysis. Core genes were defined as orthologs shared between the four *Arthrobacter* strains. Synteny was studied using an in-house developed tool, Align, using dynamic programming to search conserved gene trains allowing gaps and "mismatches" (homology relation instead of orthology). Circular representation of the genome was produced using the Circos software [27].

Un materiel et méthode décevant

Crise de la reproductibilité

- Problème général ("Reproducibility crisis")
  - Remis en avant par les science sociales (pyschologie)
  - Etendu à l'ensemble des disciplines
  - … mais un problème qui n'est pas nouveau

Et en bioinfo ?

Un problème ancien :

- En 2009, moins de la moitié des experiences de transcriptomique parues dans Nature Genetics n'ont pu être reproduites
- Sur 50 articles citant BWA en 2011, 31 ne citent ni version, ni paramètres. 26 ne donnent pas accès aux données sous-jacentes

- Selon un sondage mené auprès de plus de 1500 biologistes

    - 70% ont déjà éprouvé des difficultés à reporduire une analyse (Baker 2016)

- Ten Years Reproducibility Challenge refaire ses analyses d'il y a 10 ans …

Quelles sont les difficultés ?

- Problèmes d'accès aux données
- Problèmes d'accès aux outils

- Problèmes de paramètrage de l'analyse

    - version des outil
    - paramètrage de l'analyse

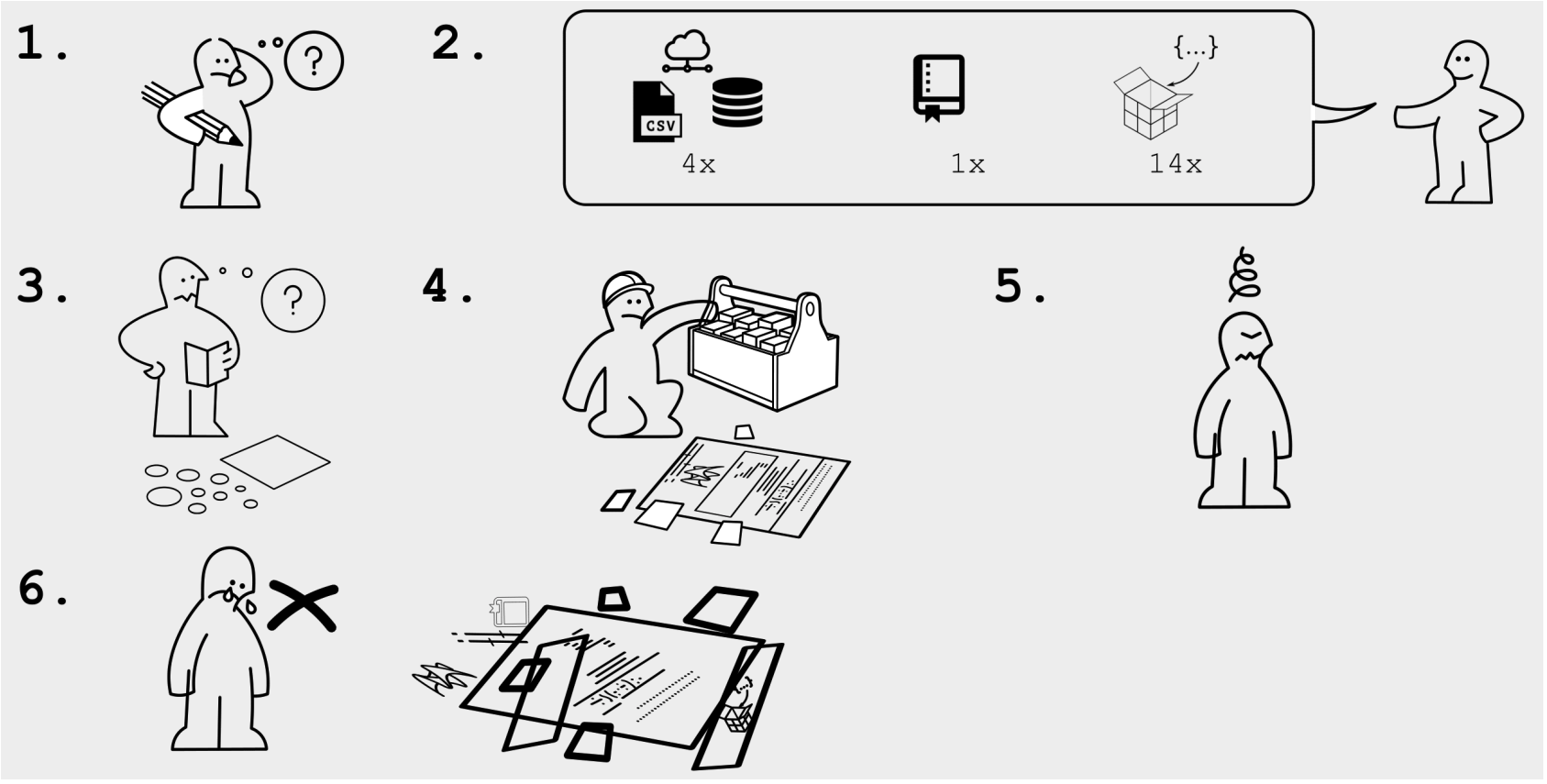
- Problème d'accès aux ressources necessaires

Replication ≠ Reproductibilité

- La replication indépendant d'analyse est à la base de la méthode scientifique
- En complément de **réplication** indépendante (experimentation, échantillonage, analyse, …), la **reproduction** d'analyse est indispensable à l'évaluation et à la compréhension de la démarche employée
- Il existe une ambiguité en anglais entre réplication (*replication*) et reproduction (*reproducibility*). Derrière la *reproducibility crisis* on mélange les deux :
    - Impossibilité de répliquer des résultats de façon indépendante (psychologie, medecine, biologie…)
    - Impossibilité de reproduire des analyses à partir des mêmes données de départ
- Chacun peut déjà, par l'utilisation d'outils conviviaux, améliorer la reproductibilité de ses travaux

Source : (Allard, 2018)

En pratique, qu'est ce qu'être reproductible ?



https://github.com/karthik/rstudio2019/blob/master/reproducible-data-analysis.pdff

En prtaique, qu'est ce qu'être reproductible (2) ?

Avoir accès :

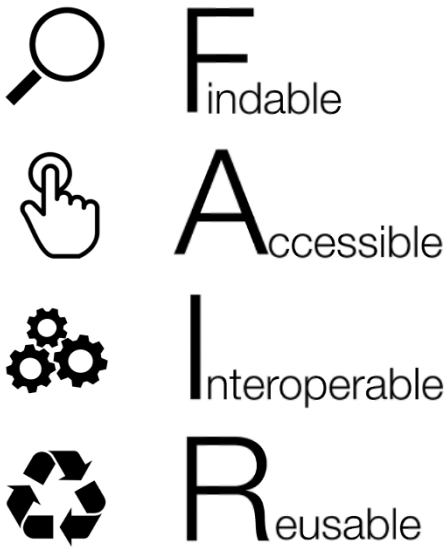
- aux pièces (les **données**)
- aux outils (les **logciels**,)

- au mode d'emploi : **paramètres**, **workflows d'analyse**

  Mais aussi :

- à la description des pièces, de la façon dont elles ont été produites (**méta-données**)
- à la docupmentation technique (**choix techniques explicites**)
- au savoir faire du monteur (**formation**)
- Eventuellement à un atelier équipé pour le montage (**ressources informatiques**)

FAIR : un pré-requis la reprouctibilité



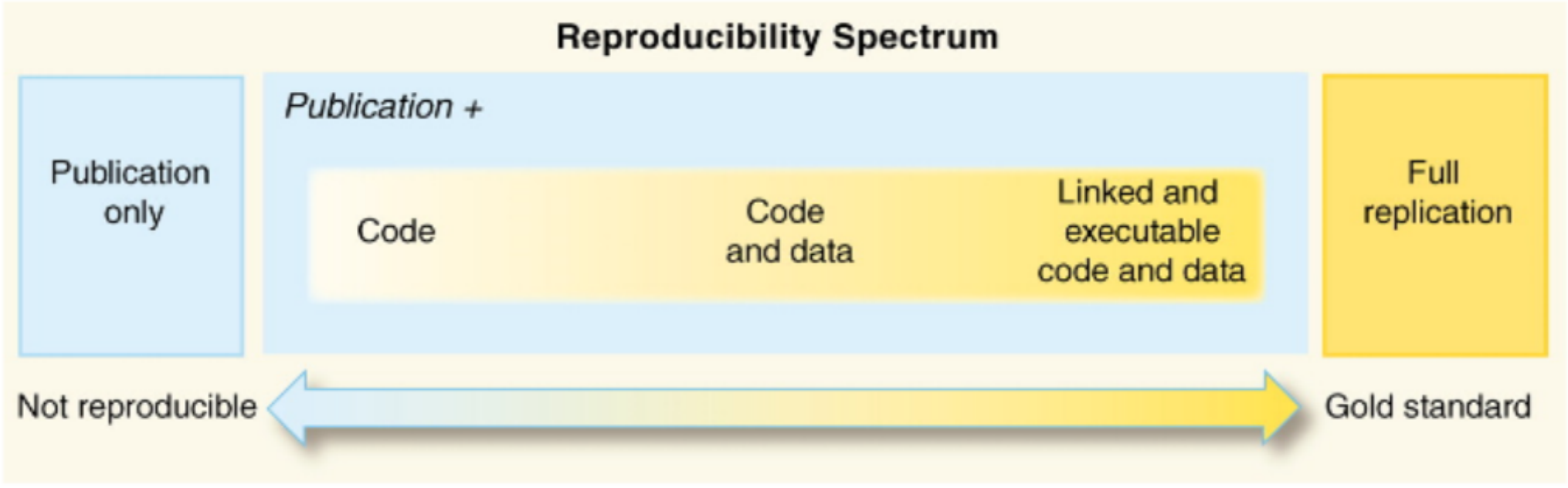
Findale Accessible Inetroprable Reusable

Principes autour des données au sens large :

- **F**acile à trouver : pour les humains et les ordinateurs :
  - id unique et perennes
  - métadonnées riches
- **A**ccessibles à **long terme**
  - entrepôt "perenne"
  - licence d'utilisation explicite (FAIR ≠ ouvert)
- **I**nteroperables : faciles à combiner avec d'autres jeux de données
  - formats ouverts et documnentés
  - vocabulaire standardisé, ontologies (données et méta-données)
- **R**éutilisables :
  - réutilisables par soi, par d'autres
  - réutilisables par des machines

(Wilkinson, Dumontier, Aalbersberg, et al., 2016)

Le spectre de la reproductibilité



Spectre de la reproductibilité,

Source : (Piazzi, Cerqueira, Manso, et al., 2018)

L'outillage

Aller de façon prgamatique vers une documentation accrue de ce que l'on fait (comment, pourquoi)

- Rendre accessible ses données à soir, aux partenaires, à tous) :
  - Data Managemlent Plan (Opidor)
  - Dépôt internationaux (ENA, NCBI)
  - DataVerse
- Définir les outils utilisés :
  - conda, bioconda
  - singularity, docker
  - Machine Virtuelle
- Décrire son workflow d'analyse, le rendre portable :
  - Galaxy
  - Snakemake, Nextflow
- Gérer les versions de ses codes, les publier :
  - git
  - github / gitlab
- Tracer son analyse dans des documents partageables et réutilisables :
  - Rmd
  - Jupyter Notebooks

Objectifs du TP

Se décomplexifier, désacraliser la reproductibilté !

Vous fournir des outils, des pistes pour rendre vos projets :

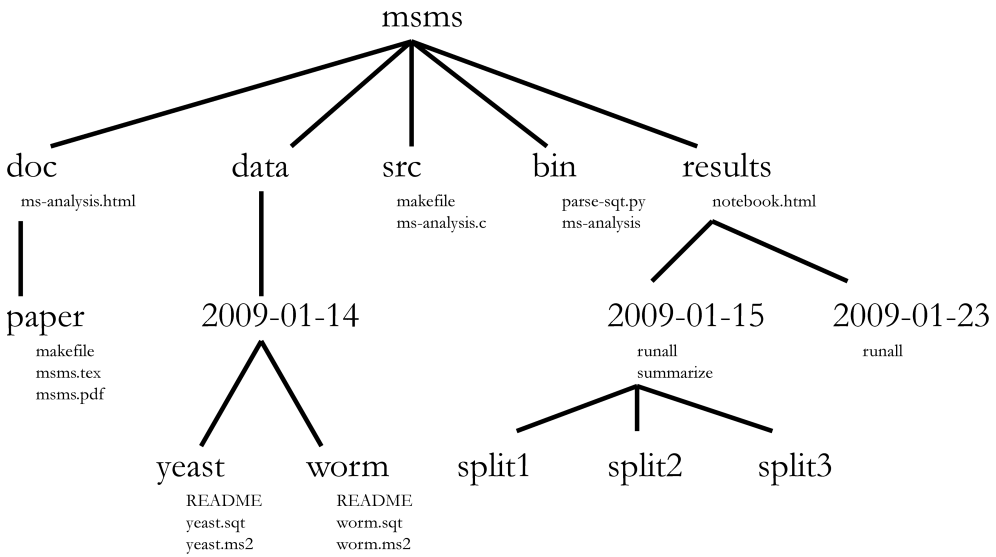
- transparents
- robustes
- réutilisables
- partageables

Bref, *plus* reproductibles.

Parties pratiques sur la versionning des documents (Git et GitHub) et les docs computationnels.

Partie 1 : Organiser son espace de travail

Partie 1 : Organiser son espace de travail



Source : (Noble, 2009)

Séparer

- données
- code
- scripts
- resultats

Organiser son espace de travail (2)

Box 1. Summary of practices

1. Data management
   a. Save the raw data.
   b. Ensure that raw data are backed up in more than one location.
   c. Create the data you wish to see in the world.
   d. Create analysis-friendly data.
   e. Record all the steps used to process data.
   f. Anticipate the need to use multiple tables, and use a unique identifier for every record.
   g. Submit data to a reputable DOI-issuing repository so that others can access and cite it.

Source : (Wilson, Bryan, Cranston, et al., 2017)

Partie 2 : Des langages à faible balisage pour faciliter la traçabilité et la prise de note

Partie 2 : Des langages à faible balisage pour faciliter la traçabilité et la prise de note

Comment mettre en forme et structurer simplement un document texte ?
→ avec un balisage faible tel que *Markdown*

- Permet :
  - Organiser les titres de sections
  - Italique / gras / souligné
  - Générer des listes
  - Ajouter des tableaux
  - Insertion d'image et de blocs de code
- Texte codé en UTF-8 (assure une pérennité, lisibilité et portabilité) facilement versionnable.
- Les langages de balisage permettent de mettre en forme convenablement le fichier pour un meilleur confort de lecture.

Exemple illustrant la simplification du balisage :

- HTML

```
<ul>
  <li>item1</li>
  <li>item2</li>
</ul>
```

- Markdown

```
- item1
- item2
```

Partie 2 : Markdown - exemples de commande

```
# Titre H1
## Sous-titre H2
### Sous-titre H3

*italique*, **gras** et `code`

> Citations

```
bloc de code
```

* liste
  * item
  * item

[lien](https://fr.wikipedia.org)
 #lien vers une image en ligne ou dans l'espac
```

Gardez ce mémo à porté de main !

Partie 3 : Versionner ses documents

Partie 3 : Versionner ses documents

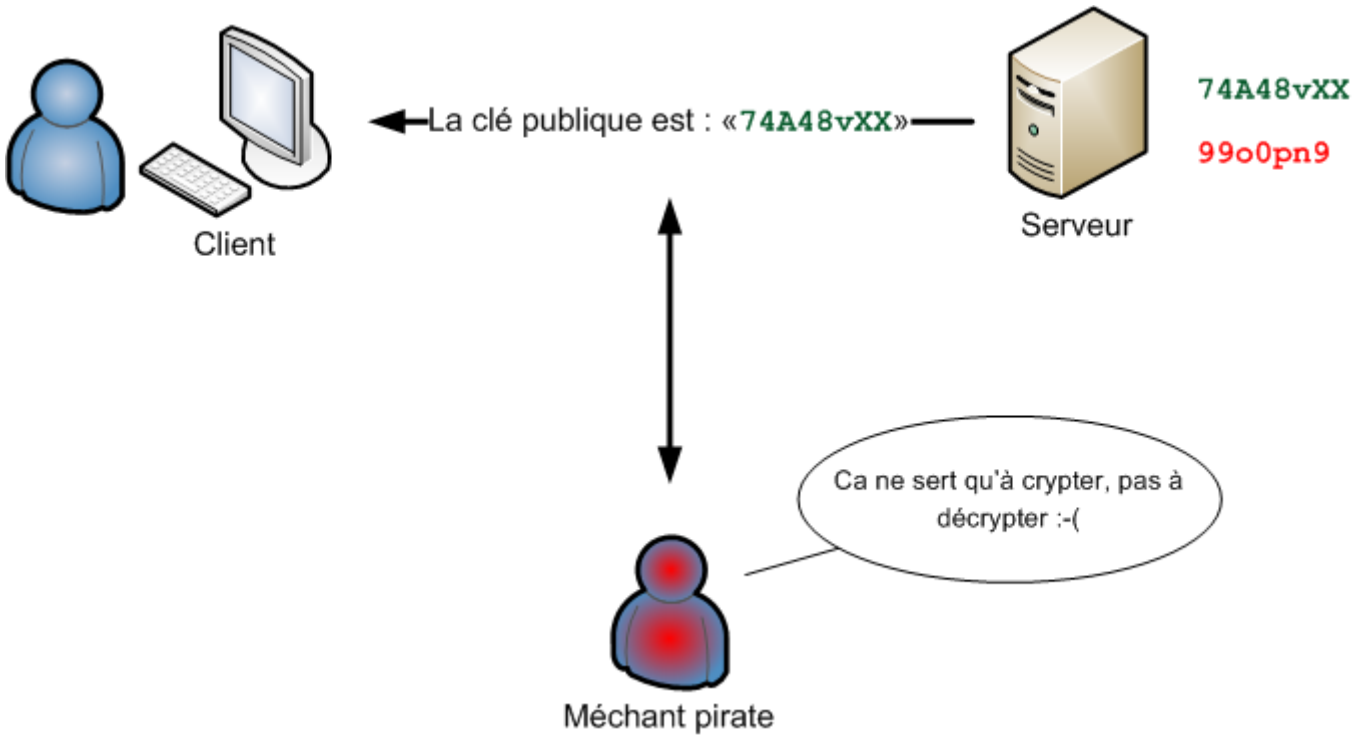
- Les documents évoluent, il est nécéssaire de suivre les versions
  - On trace toutes les modifications faites
  - On garde chaque version des documents du dossier de travail
  - C'est un peu comme copier-coller son dossier de travail … mais en beaucoup plus précis et pratique !
  - `git` est un standard dans la gestion des versions distribuée

- **Repository** = Dossier / Projet
- **Commit** = Enregistrement d'un ensemble de fichier à un instant T (= photo)
- **Branche** = Ensemble chainé de commits, par défaut la branche principale s'appelle « master » (cette notion n'est pas primordiale pour débuter)

Voici un aide mémoire git

Retours sur les clés SSH

- Protocole pour la sécurisation des transferts de données.
- Méthode de chiffrement assymétrique qui fonctionne avec une paire de clé :
  - une clé *publique* qui sert à chiffrer (et que vous pouvez partager à l'extérieur)
  - une clé *privée* qui sert à déchiffrer (et que vous gardez précieusement secrète)
- Permet d'établir un tunnel sécurisé entre deux machines


La clé publique est : «74A48vXX»
74A48vXX
99o0pn9
Client
Serveur
Ca ne sert qu'à crypter, pas à décrypter :-(
Méchant pirate

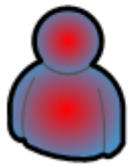
*Crypte une clé symétrique de son choix (**topsecret**) avec la clé publique « **74A48vXX** »*

Client

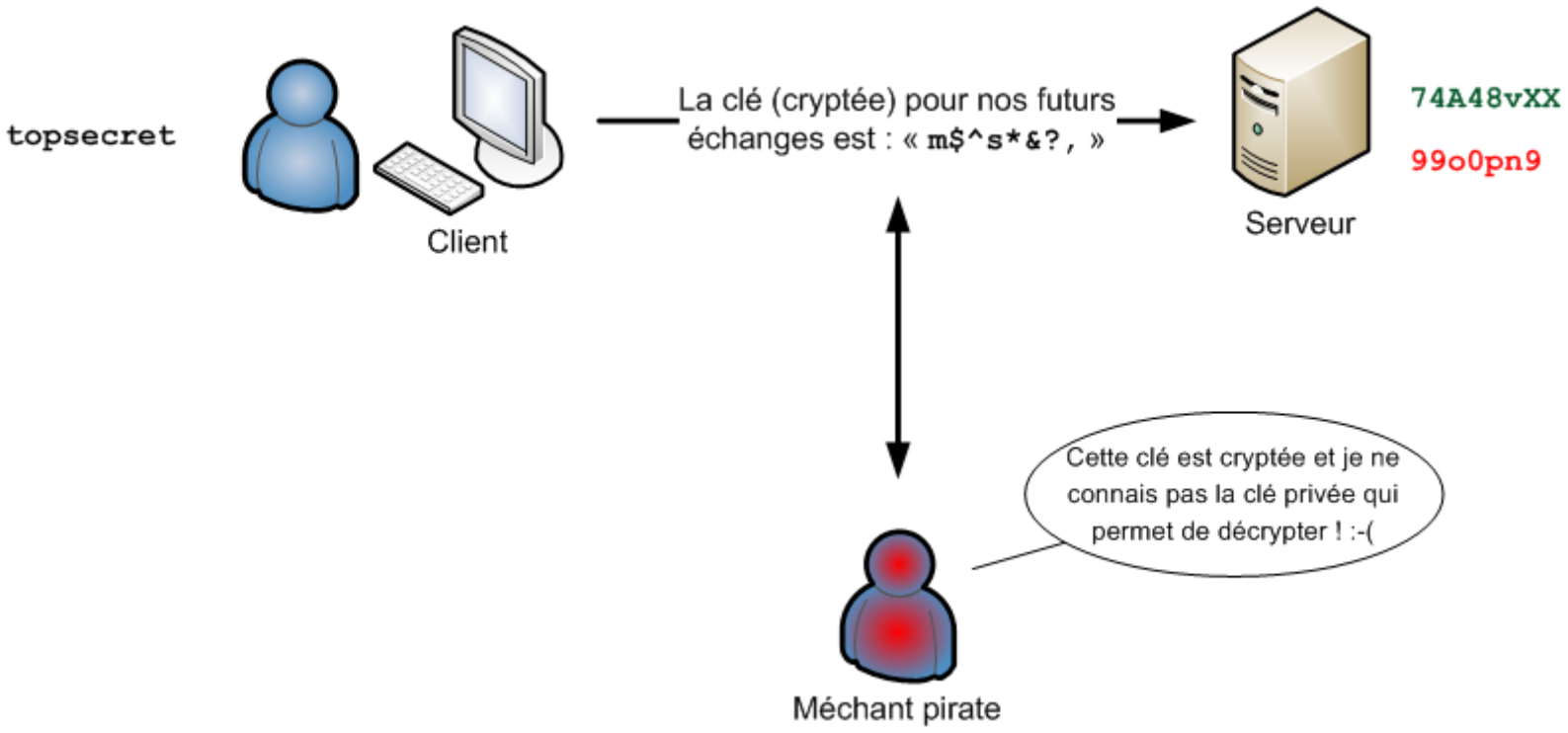
Serveur

74A48vXX

99o0pn9



Méchant pirate


topsecret
Client
La clé (cryptée) pour nos futurs échanges est : « m$^s*&?, »
Serveur
74A48vXX
99o0pn9
Cette clé est cryptée et je ne connais pas la clé privée qui permet de décrypter ! :-(
Méchant pirate

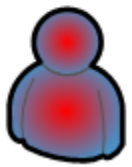
`topsecret`

Client

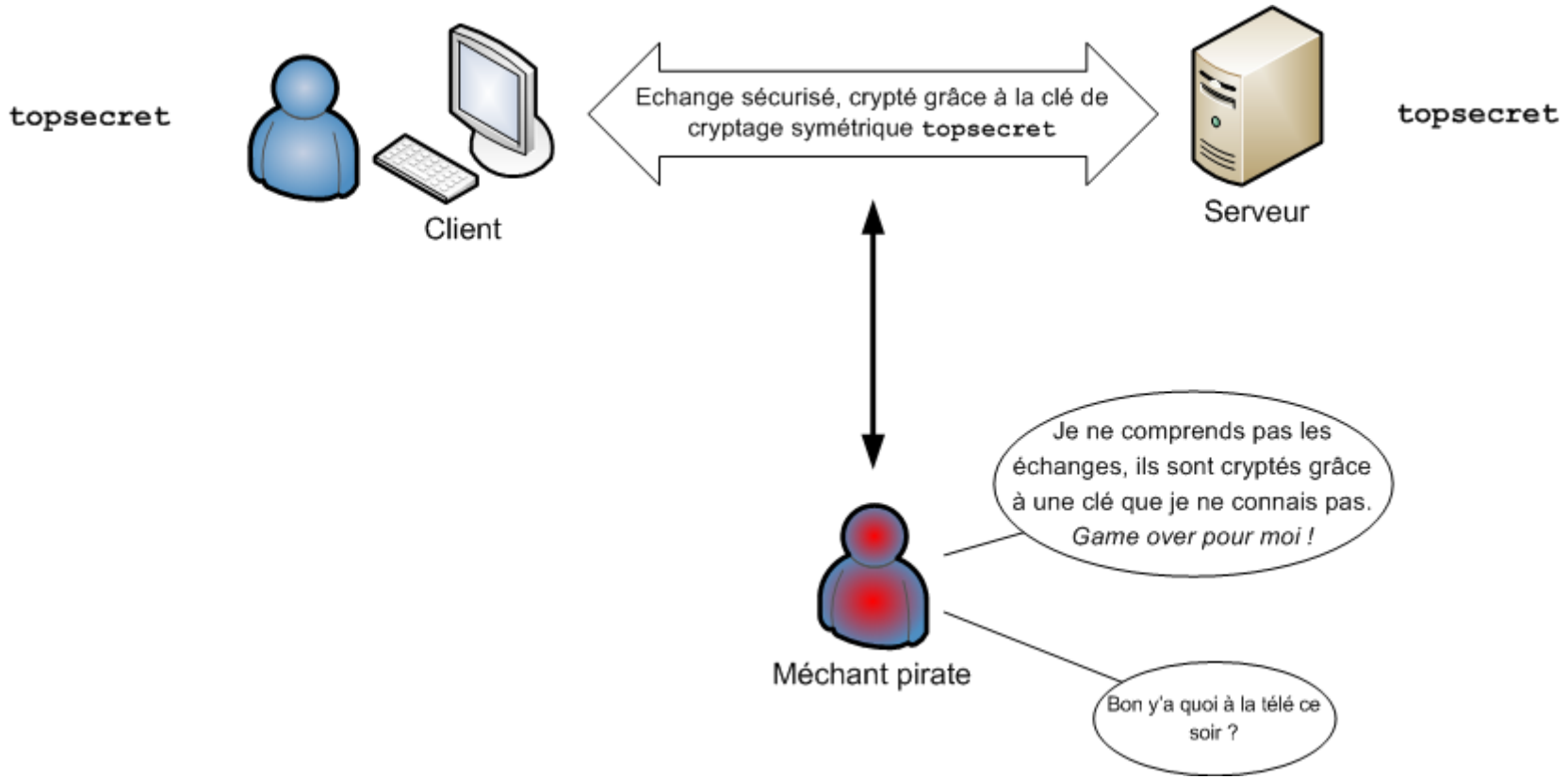
Serveur

*Décryptage de « `m$^s*&?,` » avec la clé privée « **99o0pn9** »*

La clé pour le cryptage symétrique est `topsecret` !



Méchant pirate


topsecret
Client
Echange sécurisé, crypté grâce à la clé de cryptage symétrique topsecret
Serveur
topsecret
Je ne comprends pas les échanges, ils sont cryptés grâce à une clé que je ne connais pas. Game over pour moi !
Méchant pirate
Bon y'a quoi à la télé ce soir ?

Retours sur les clés SSH

- Protocole pour la sécurisation des transferts de données.
- Méthode de chiffrement assymétrique qui fonctionne avec une paire de clé :
  - une clé *publique* qui sert à chiffrer (et que vous pouvez partager à l'extérieur)
  - une clé *privée* qui sert à déchiffrer (et que vous gardez précieusement secrète)
- Permet d'établir un tunnel sécurisé entre deux machines

- On utilise `ssh-keygen -t rsa` pour générer la paire de clé
- Puis `ssh-copy-id -i id_rsa.pub <login>@<server>` pour envoyer la clé publique
- Maintenant on peut se connecter de manière sécurisée via `ssh <login>@<server>` sans renseigner de clé
- Pour GitHub, on renseigne les clés publiques via l'interface graphique

TP : Utilisation de Git

1. Initialiser le dépôt (en local)

```
mkdir testRepo
cd testRepo
git init
```

2. Ajouter un document (en local)

```
touch firstFile
emacs firstFile

git status
```

TP : Utilisation de Git

3. Versionner un document (en local)

```
git add firstFile
git commit firstFile

git status
```

4. Workflow de modification (en local)

```
emacs firstFile

git status
git diff

git commit -m "modification"
```

TP : Utilisation de GitHub

5. Initialisation du dépôt distant (Github)

- Créer un dépôt distant sur Github

6. Lier le dépôt local et disatnt

```
git remote add origin git@github.com:vloux/myFirstRepo.git
git remote -v
```

7. Pousser les modifications locales sur le dépôt distant

```
git push origin master
```

TP : Utilisation de GitHub

8. Verification sur Github des infos

TP : Utilisation de GitHub

9. Modification du dépôt par l'interface

- ajouter un `README.md` (avec markdown)

TP : Utilisation de Git et GitHub

8. Récupérer sur le repo local des infos de la branche distante

```
git fetch origin master
git merge origin master
```

équivalent à :

```
git pull origin master
```

TP : Utilisation de Git et GitHub

Nommer des versions (tags)

```
git tag version1 -m "Initial version"

git tag

git push origin master --tags
```

TP : utilisation de Git et GitHub

Comparer à une version précédente

```
git log

git diff [CommintNumber] aNewFile
```

En résumé

- `git clone` : cloner un dépot distant
- `git init` : initialiser le versionning sur un dépot local
- `git commit` : enregister l'état d'un dépôt
- `git status` : afficher l'état des documents du dépôt
- `git diff` : comparer l'état actuel au dernier commit, ou deux commits entre eux ou deux branches

et bien d'autres encore (`blame`, revert`,...)

CheatSheet

Pour apprendre progressivement et/ou se simplifier la vie

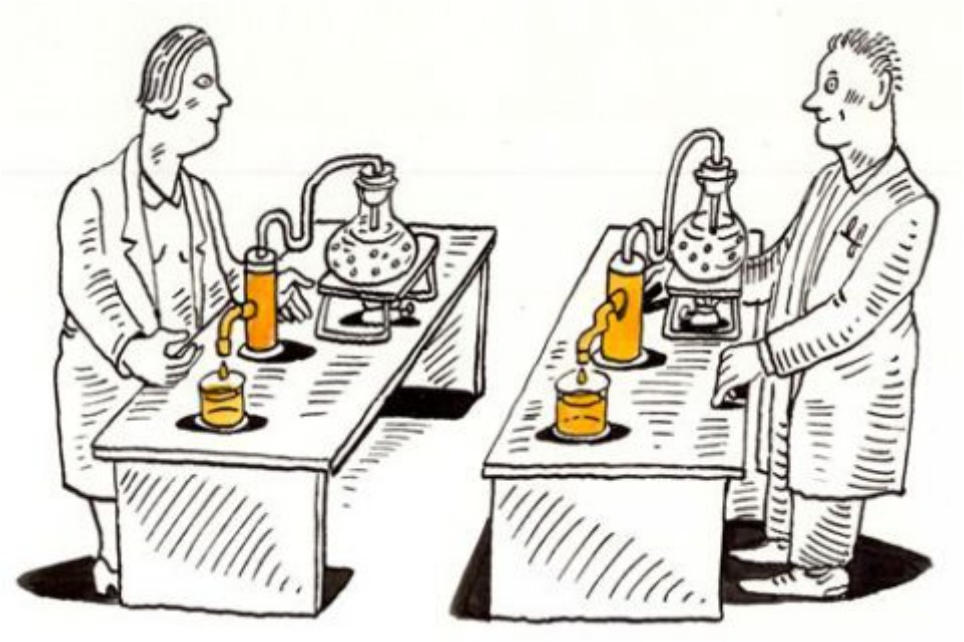
- Github Desktop
- Les intégration aux différents IDE (RStudio and co)


COFFEE
BREAK

Partie 4 : Documents computationels - Notebook

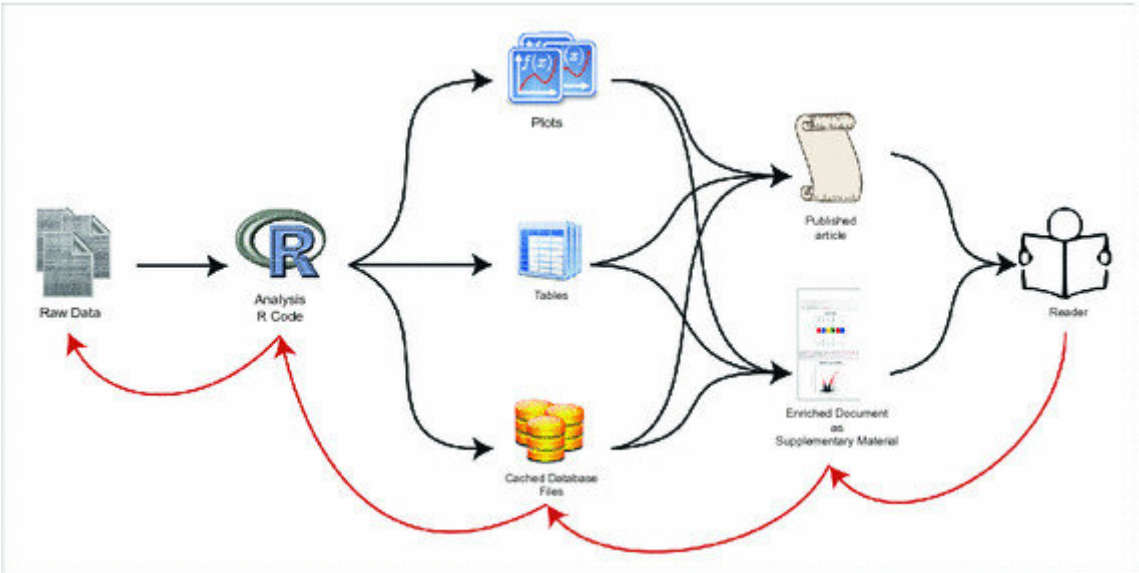
Partie 4 : Documents computationels - Notebook

- Il faut se donner les moyens pour qu'autrui puisse inspecter nos analyses
- Expliciter pour augmenter les chances de trouver les erreurs et de les éliminer
  - Inspecter pour justifier et comprendre
  - Refaire pour vérifier, corriger et réutiliser



Partie 4 : Documents computationels - Notebook

- Regrouper dans *un unique document*:
  - Les informations, le code, calculs et les résultats
  - Pour assurer leur cohérence et améliorer la traçabilité.
  - Exportable (ex : html) pour une meilleure portabilité et lisibilité.



(Russo, Righelli, and Angelini, 2016)

Encore un joli mémo pour R markdown.

TP : Rédaction d'un notebook avec RStudio

1. Connectez vous au RStudio de l'IFB (https://rstudio.cluster.france-bioinformatique.fr/).
2. Clonez le projet :
    - New Project (en haut à droite)
    - Version Control
    - Git
    - Repository URL (clone with SSH : `git@github.com: ...`)
3. Explorez le dépot depuis ce 3e device remote
4. Créez un document R Markdown :
    - File
    - New file
    - R Markdown

Dans ce document on a :

- Une entête générale
- Du texte, mis en forme avec markdown
- Du code R dans des chunks
- Des résultats et outputs

TP : Rédaction d'un notebook avec RStudio

- Grace à l'onglet git de Rstudio, vous pouvez suivre l'état des fichiers, commit, diff, push/pull, ...
- Chaque chunk peut être exécuté individuellement grâce à la flèche verte.
  Les options assosiées à chaque chunks sont disponible avec la roue crantée.

1. Modifier le document (en plusieurs commit)
   - ajoutez un chunk `knitr::kable(head(iris))` pour visualiser un table
   - ajoutez un chunk `plot(cars)` pour un plot
   - ajoutez des commentaires
   - puis commitez les modifications.
2. Lorsque vous êtes satisfait de votre raport, générez la version HTML en cliquant sur Knit. Commitez, pushez, ...
3. Visualisez les modifications côté GitHub.

Votre rapport est disponible, versionné et partageable !

TP : Partager un document html avec GitHub Pages

- Dans les options du repo, dans le chapitre "GitHub Pages", activer la source correspondant à la branch master.
- la page est disponible à l'adresse https://[user].github.io/[repo]/[page].html

Suivant l'interlocuteur, partagez le `.Rmd` ou le `.html`

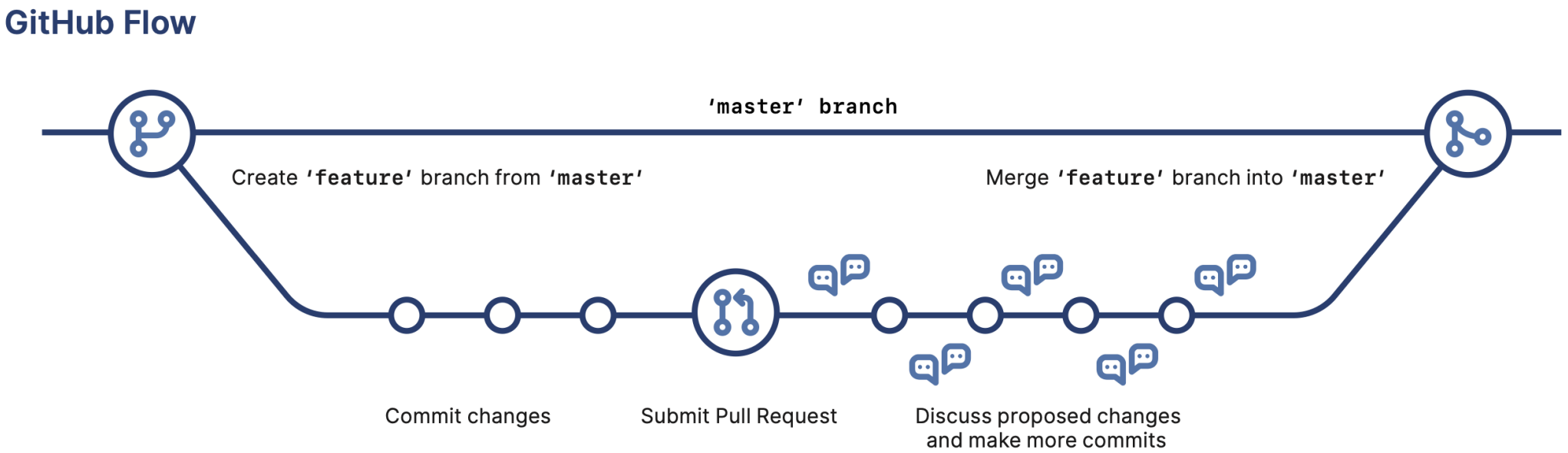
Conclusion :

La reprouctibilité, comme la "FAIRisation" sont des processus.

Des bonnes pratiques appuyées par des outils qui les facilitent

- Organiser ses analyses
- Décruire correctement ses données et ses processus d'analyse (PGD, FAIR)
- Tracer ses analyses à l'aide de documents computationnels (Rmd, Jupyter Notebooks, …) :
  - Transparents
  - Accessibles
  - Partageables
- Versionner ses documents computationnels (GitHub, GitLab, …)
  - Traçabilité
  - Accessibilité

Aller plus loin - travailler en commun avec Git et Github :



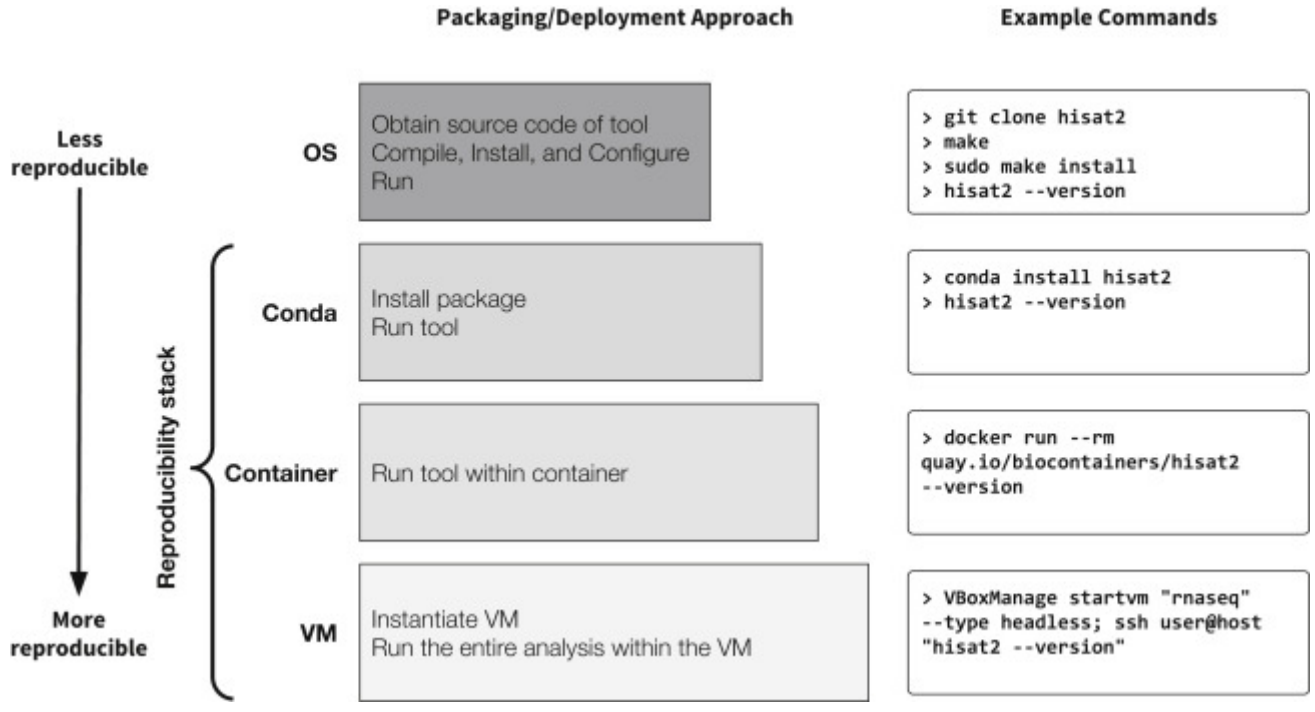
- Branche : version parallèle à la version principale
- Pull Request : demande de fusion des modifications d'une brache vers la branche principale

Un exemple de PR

Pour aller + loin - Fixer et partager son environnement :

- Conda et Bioconda
  - gestion des dépendances, versions
  - Possibilité de créer un environnement par analyse
  - Exporter son environnement dans un fichier `env.yml` et le versionner
  - `conda env export > environment.yml`
- Containers, machines virtuelles
  - Docker, Singularity, VM virtualbox
  - Pour les outils non "condaoisables", les environnements complexes
  - Les images Singularity sont déployables sur les infrastructures type IFB et s'éxecutent "presque" comme un executable

Pour aller plus loin - Fixer et partager son environnement (2)



(Grüning, Chilton, Köster, et al., 2018)

Pour aller + loin - gestionnaires de workflows

SnakeMake, NextFlow pour :

- Définir de façon "simple" et modulaire des workflows d'analyse :
  - Parallelisables : les étapes indépendantes peuvent être jouées en parallèle.
  - Qui assurent la reprise sur erreur : si on refait une analyse, change un paramètre, seul ce qui doit être rejoué est relancé.
  - Portables : un même script peut être joué en local, sur des clusters différents en changeant le fichier de configuration.
  - Partageables : un fichier texte versionné
  - Peut gérer pour vous le versionning et l'installation des outils avec Conda

Pour aller + loin - exemple de Snakefile

Bash

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

Snakefile

```
SAMPLES, = glob_wildcards("./{smp}.fastq.gz")

rule final:
  input:expand("fastqc/{smp}/{smp}_fastqc.zip",smp=SAMPLES)
rule fastqc:
  input: "{smp}.fastq.gz"
  output: "fastqc/{smp}/{smp}_fastqc.zip"
  message: """Quality check"""
  shell: """fastqc {input} --outdir fastqc/{wildcards.smp}"""
```

Pour aller + loin - FAIRifier ses données

Dépots dans les dépôts publics :

- Dans les dépôts thématiques
  - données brutes
  - données analysées
  - /!\ méta-données

- dans les dépots généralistes (dataverse , figshare, …)

  - fichiers tabulés; "autres" données. ce qu'on mettrait en suypplementary material.
  - (éventuels) leins vers les fichiers de données

- Publier un data-paper ?

Pour aller + loin :

- jusqu'où aller dans la reproductibilité ?
  - Mat et Met électroniques :
  - galaxy Pages
  - Gigascience, GigaDB
- Quel est le rapport coût / bénéfice ?

Ressources

- FUN MOOC Recherche Reproductible
- FAIR Bioinfo
- Cours Git et Github
- Github pages
- Rmd the definitive Guide
- Snakemake
- NextFlow et nf-core
-

- Les mémo présentés dans ce cours :
  - markdown
  - git
  - R markdown

References

Allard, A. (2018). *La crise de la réplicabilité*. URL: https://laviedesidees.fr/La-crise-de-la-replicabilite.html.

Grüning, B, J. Chilton, J. Köster, et al. (2018). "Practical Computational Reproducibility in the Life Sciences". In: *Cell Systems* 6.6, pp. 631-635. DOI: 10.1016/j.cels.2018.03.014. URL: https://doi.org/10.1016/j.cels.2018.03.014.

Noble, W. S. (2009). "A Quick Guide to Organizing Computational Biology Projects". In: *PLOS Computational Biology* 5.7, pp. 1-5. DOI: 10.1371/journal.pcbi.1000424. URL: https://doi.org/10.1371/journal.pcbi.1000424.

Piazzi, A. C, A. S. Cerqueira, L. R. Manso, et al. (2018). "Reproducible research platform for electric power quality algorithms" , pp. 1-6.

Russo, F., D. Righelli, and C. Angelini (2016). "Advantages and Limits in the Adoption of Reproducible Research and R-Tools for the Analysis of Omic Data". In: Vol. 9874. , pp. 245-258. DOI: 10.1007/978-3-319-44332-4_19.

Wilkinson, M. D, M. Dumontier, I. J. Aalbersberg, et al. (2016). "The FAIR Guiding Principles for scientific data management and stewardship". In: *Scientific Data* 3.1. DOI: 10.1038/sdata.2016.18. URL: https://doi.org/10.1038/sdata.2016.18.

Wilson, G, J. Bryan, K. Cranston, et al. (2017). "Good enough practices in scientific computing". In: *PLOS Computational Biology* 13, pp. 1-20. DOI: 10.1371/journal.pcbi.1005510. URL: https://doi.org/10.1371/journal.pcbi.1005510.