Bonnes pratiques pour les projets de code

Des pratiques et des outils qui peuvent grandement améliorer la qualité de nos projets de code

Romain Avouac, Lino Galiana

La notion de bonnes pratiques

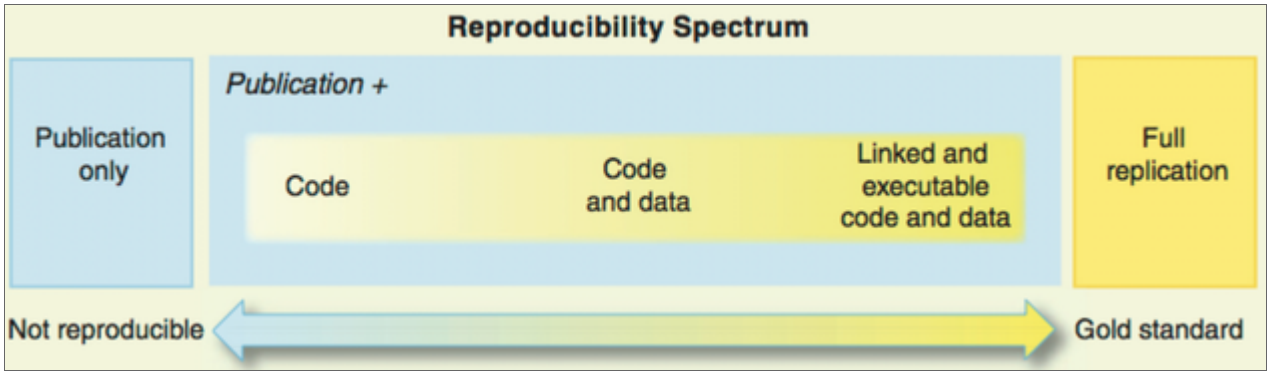
Contexte

- **Origine** : communauté des développeurs logiciels
- **Constats** :
  - le code est plus souvent lu qu'écrit
  - la maintenance d'un code est très coûteuse
- **Conséquence** : ensemble de **règles informelles**, conventionnellement acceptées comme produisant des logiciels **fiables**, **évolutifs** et **maintenables**

Pourquoi s'intéresser aux bonnes pratiques ?

- L'activité du statisticien / data scientist tent à se rapprocher de celle du développeur :
  - projets **intenses en code**
  - **projets collaboratifs** et de grande envergure
  - **complexification** des données et donc des **infrastructures**
  - **déploiement** d'applications pour valoriser les analyses

Bonnes pratiques et reproductibilité



- Le manque de reproductibilité a des effets négatifs concrets (cf. Krugman : "The Excel Depression")...
- et réduit les chances qu'un projet passe en prod

Un socle de bonnes pratiques ?

- Nécessité d'**arbitrer** entre :
  - Le **gain de qualité** permis par l'adoption des bonnes pratiques
  - Le **coût** élevé d'une reproductibilité parfaite
- L'arbitrage pertinent se décide au **niveau du projet**
- Peut-on s'orienter vers un **socle commun** de bonnes pratiques ?

Un socle de bonnes pratiques

Normes de qualité du code

- Enjeu : produire un code **lisible** et **maintenable**
- Des **principes généraux** : lisibilité, concision, cohérence, documentation
- Des **standards communautaires** (`R` : `Tidyverse` guide, `Python` : PEP 8)
- Des outils : **linters** et **formatters**

Normes de structure des projets

- Enjeu : produire des projets **lisibles**, **maintenables** et **reproductibles**
- Des **principes généraux** :
  - Données brutes immuables
  - Modularité
  - Séparation stockage / code / configuration
  - Structure de packages
- Des **outils / standards communautaires** : **templates de projets** (R : RProjects, Python : cookiecutters)

Contrôle de version

- Enjeux du **contrôle de version** :
  - Construire l'**historique** de son projet
  - **Documenter en continu** son projet
  - Favoriser le **travail collaboratif**
- Un outil incontournable : **`Git`**

Aller plus loin

La qûete (infinie) de la reproductibilité

- En fonction des **spécificités** du projet, d'autres bonnes pratiques ;
- Viser la **portabilité** du code:
  - Environnements reproductibles
  - Conteneurisation (`Docker`)
- **Automatiser** ce qui peut l'être:
  - Intégration continue

La qûete (infinie) de la reproductibilité (2/2)

- Mettre à disposition différents **livrables** ;
- Pose question **déploiement**:
  - modèle sous forme API,
  - rapport reproductible sous forme site web
  - ...

Discussion

- Ce socle de bonnes pratiques vous semble-t-il pertinent ?
- Les bonnes pratiques vous ont-elles été enseignées ou vous les avez acquises par expérience ?
- Une lecture qui vous a inspiré sur le sujet ?
- Comment diffuser, former et inciter à l'utilisation des bonnes pratiques ?
- Un outil de prédilection pour mettre en oeuvre les bonnes pratiques ?

Ressources

- **https://ensae-reproductibilite.netlify.app/**
- **https://eliocamp.github.io/reproducibility-with-r/**
- **https://mitmat.github.io/slides/2022-05-26-egu/code-data-open-science.html#1**
- **https://www.pratiques.utilitr.org/** (à actualiser)