

# HMMA238: Introduction à git

**Joseph Salmon**

<http://josephsalmon.eu>

Université de Montpellier



# Sommaire

Motivation

Démarrer

# Sommaire

Motivation

Démarrer

## Crédit:

**Inspiré du travail de Nelle Varoquaux:**

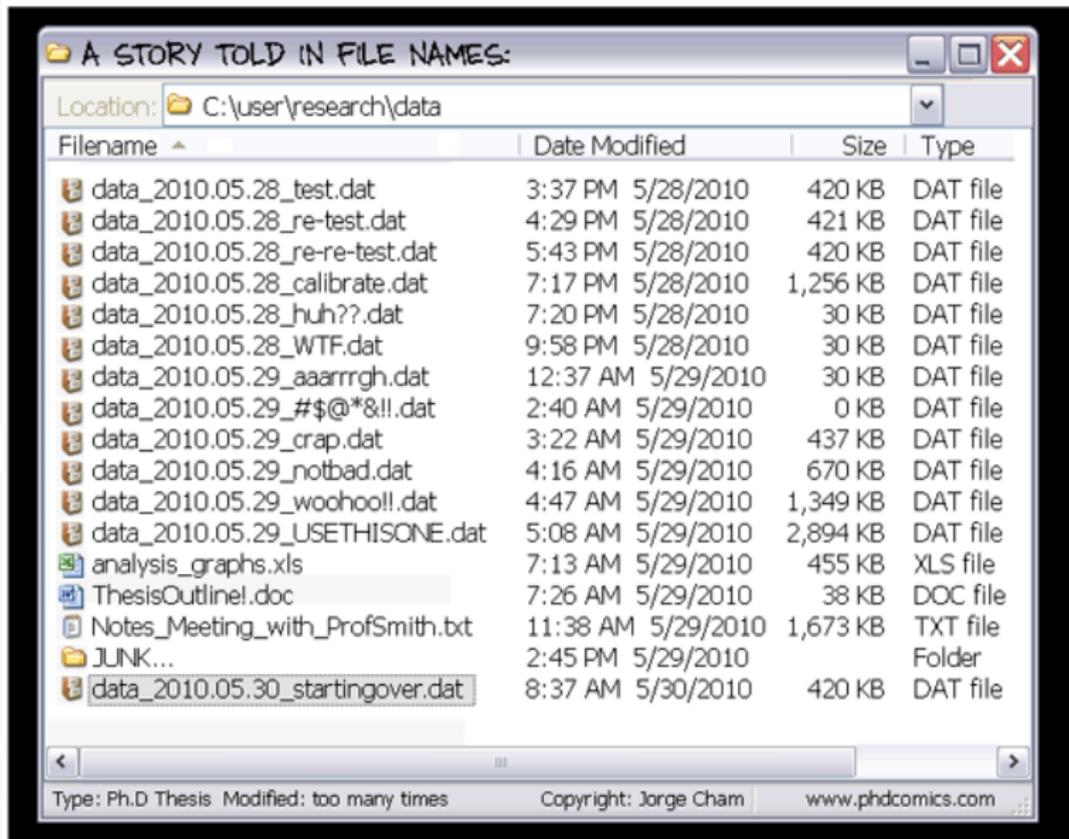
<http://members.cbio.mines-paristech.fr/~nvaroquaux/teaching/2016-aspss/git.html>

# Pourquoi utiliser un gestionnaire de version

- ▶ recherche / travail reproductible
- ▶ retrouver un “état” passé (typiquement un état où tout marchait bien)
- ▶ recherche / travail rapide et efficace (notamment à plusieurs)

## Systeme similaires (limités?)

- ▶ dropbox, solutions “cloud” et variantes
- ▶ Sharelatex (bon en fait on peut l'utiliser avec git aussi)



Credit: "Piled Higher and Deeper" by Jorge Cham: [www.phdcomics.com](http://www.phdcomics.com)

# Contrôle de version

- ▶ garde **TOUT** l'historique des versions pour un traçage facile
- ▶ bénéficie au travail d'équipe<sup>(1)</sup>
- ▶ Améliore l'efficacité d'écriture (de code ou de texte)

---

<sup>(1)</sup> "Je suis une bande de jeune à moi tout seul" (R. Séchan, 1977)

# Visualisation: exemples avec github

- ▶ Un petit projet personnel:  
<https://github.com/josephsalmon/OrganizationFiles>
  
- ▶ Un gros projet:  
<https://github.com/scikit-learn/scikit-learn>

# Exercices

---

**Exo:** Commencez par vous créer un compte github si ce n'est pas déjà le cas

---

---

**Exo:** Vérifier que git est installée sur votre machine, sinon:  
(Linux): `sudo apt-get git`

---

# Sommaire

Motivation

Démarrer

# Créer un projet

- ▶ Initialisation : nouveau projet

```
$ git init  
Initialized empty Git repository in /home/jo/Downloads/CES/.git/
```

- ▶ Initialisation : projet existant

```
$ git clone git@github.com:josephsalmon/OrganizationFiles.git
```

# Configurer git

- ▶ Localement : seulement le répertoire git courant sera affecté

```
$ git config [options]
```

- ▶ Pour l'utilisateur courant : la configuration de l'utilisateur sera modifié dans le fichier `/.git/config`

```
$ git config --global [options]
```

- ▶ Globally: tous les utilisateurs du système seront affectés

```
$ git config --system [options]
```

# Configurer git (II)

- ▶ Identité :

```
$ git config --global user.name "Your Name Comes Here"  
$ git config --global user.email you@yourdomain.com
```

- ▶ Pour choisir l'éditeur (qui lira les différences)

```
$ git config --global core.editor vim
```

- ▶ Vérifier son système

```
$ git config --list
```

# Commandes fondamentales

- ▶ `git add`: ajoute un fichier à la "capture" souhaitée

```
$ git add README
```

- ▶ `git commit`: sauvegarde tous les fichiers ajoutés dans la capture:

```
$ git commit -m "Mon message de commit"
```

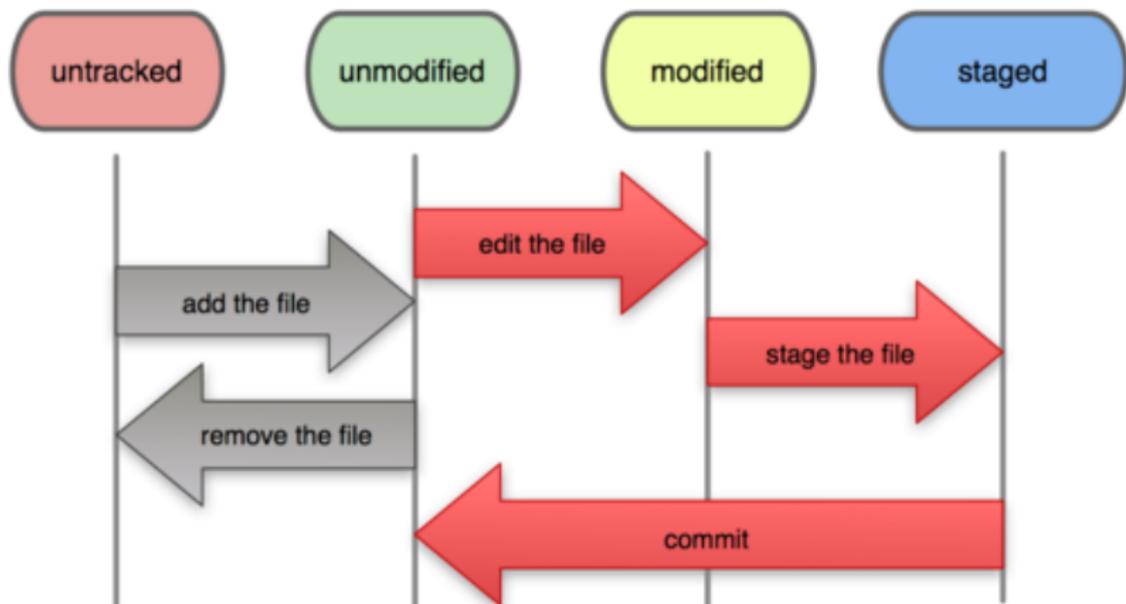
- ▶ `git status`: montre le statut des fichiers du dépôt
- ▶ `git log`: montre les logs des commits

# Status des fichiers

- ▶ fichiers tracés ( : *tracked files*)
  - ▶ non modifié ( : *unmodified*)
  - ▶ modifié ( : *modified*)
  - ▶ organisé ( : *staged*)
  
- ▶ fichiers non tracés ( : *untracked files*): tout le reste

# Cycle de vie des fichiers versionnés

## File Status Lifecycle



Crédit: <https://git-scm.com/book/en/v2>

# Exercice

---

**Exo:** Faire `01_configuring_and_committing.rst` dans

<http://members.cbio.mines-paristech.fr/~nvaroquaux/teaching/2016-aspss/exercises.zip>

---

# Effacer / déplacer des fichier

- ▶ `git mv`: déplace / renomme un fichier

```
$ git mv FILENAME TARGET
```

# Branches

- ▶ `git branch`: gère les branches
  - ▶ `git branch`: liste les branches du dépôt locale
  - ▶ `git branch [branch_name]`: créer une branche
  - ▶ `git branch -d [branch_name]`: efface une branche
  
- ▶ `git checkout`: gère les déplacements
  - ▶ `git checkout [branch_name]`: bouge la branche
  - ▶ `git checkout -b [branch_name]`: crée et déplace la branche `branch_name`

# Serveur distant

Ici github, ou un autre serveur rentre en jeu, typiquement avec la commande:

```
$ git clone git@github.com:josephsalmon/OrganizationFiles.git
```

- ▶ `git push [remote-name] [branch-name]`: pousse `[branch-name]` sur le serveur distant et la branche `[branch-name]`
- ▶ `git pull [remote-name] [branch-name]`: tire `[branch-name]` depuis le serveur distant `[branch-name]`