

# Manipuler Git avec Rstudio

Annaig De-Walsche, Marina Gomtsyan, Mary Savino

AgroParisTech

Novembre 2022



## Objectif de la séance



# Principe de Git

- ▶ Git est un gestionnaire de version. Il gère l'historique d'un ensemble de fichiers contenus dans un répertoire : "dépôt".
- ▶ Chaque fichier déclaré par l'utilisateur est suivi, et à chaque "commit", les modifications apportées aux fichiers suivis sont enregistrées.
- ▶ Fonctionnalités de Git :
  - ▶ Revenir à une version antérieure
  - ▶ Travailler à plusieurs
  - ▶ Travailler en parallèle
  - ▶ Simplifier la réconciliation entre deux versions différentes

## Le workflow de Git



1. dépôt de travail (*working directory*) contient les fichiers réels
2. l'index (*Index Stage*) agit comme une zone de transit
3. le **HEAD** pointe vers le dernier commit effectué

## Les commandes principales

- ▶ `git init`

Utilisé pour construire un nouveau dépôt

**Utilisation** : `git init [nom du dépôt]`

## Les commandes principales

- ▶ `git init`

Utilisé pour construire un nouveau dépôt

**Utilisation** : `git init [nom du dépôt]`

- ▶ `git clone`

Utilisé pour obtenir une copie d'un dépôt existant

(local/distant)

**Utilisation** : `git clone [/chemin/vers/dépôt] (local)`

`git clone [username@host:/chemin/vers/dépôt]`  
(*distant*)

## Les commandes principales

▶ `git init`

Utilisé pour construire un nouveau dépôt

**Utilisation** : `git init [nom du dépôt]`

▶ `git clone`

Utilisé pour obtenir une copie d'un dépôt existant  
(local/distant)

**Utilisation** : `git clone [/chemin/vers/dépôt] (local)`

`git clone [username@host:/chemin/vers/dépôt]`  
*(distant)*

▶ `git add`

Utilisé pour ajouter un fichier à la zone de transit (Index)

**Utilisation** : `git add [nom du fichier]`

## Les commandes principales

### ▶ `git init`

Utilisé pour construire un nouveau dépôt

**Utilisation** : `git init [nom du dépôt]`

### ▶ `git clone`

Utilisé pour obtenir une copie d'un dépôt existant

(local/distant)

**Utilisation** : `git clone [/chemin/vers/dépôt] (local)`

`git clone [username@host:/chemin/vers/dépôt]`  
(distant)

### ▶ `git add`

Utilisé pour ajouter un fichier à la zone de transit (Index)

**Utilisation** : `git add [nom du fichier]`

### ▶ `git commit`

Utilisé pour commiter dans le HEAD, mais pas encore dans le dépôt distant

**Utilisation** : `git commit -m "Message de commit"`



## Les commandes principales

- ▶ `git status`

Cette commande liste tous les fichiers qui sont en attente de commit.

**Utilisation** : `git status`

## Les commandes principales

- ▶ `git status`

Cette commande liste tous les fichiers qui sont en attente de commit.

**Utilisation** : `git status`

- ▶ `git rm`

Utilisé pour supprimer un fichier à la fois dans le *working directory* et dans Index. Pour affecter cette suppression à HEAD, il faut par la suite commiter

**Utilisation** : `git rm [nom du fichier]`

## Les commandes principales

- ▶ `git status`

Cette commande liste tous les fichiers qui sont en attente de commit.

**Utilisation :** `git status`

- ▶ `git rm`

Utilisé pour supprimer un fichier à la fois dans le *working directory* et dans Index. Pour affecter cette suppression à HEAD, il faut par la suite commiter

**Utilisation :** `git rm [nom du fichier]`

- ▶ `git pull`

Cette commande récupère et fusionne les changements du dépôt distant dans le *working directory*

**Utilisation :** `git pull`

## Les commandes principales

- ▶ `git status`

Cette commande liste tous les fichiers qui sont en attente de commit.

**Utilisation** : `git status`

- ▶ `git rm`

Utilisé pour supprimer un fichier à la fois dans le *working directory* et dans Index. Pour affecter cette suppression à HEAD, il faut par la suite commiter

**Utilisation** : `git rm [nom du fichier]`

- ▶ `git pull`

Cette commande récupère et fusionne les changements du dépôt distant dans le *working directory*

**Utilisation** : `git pull`

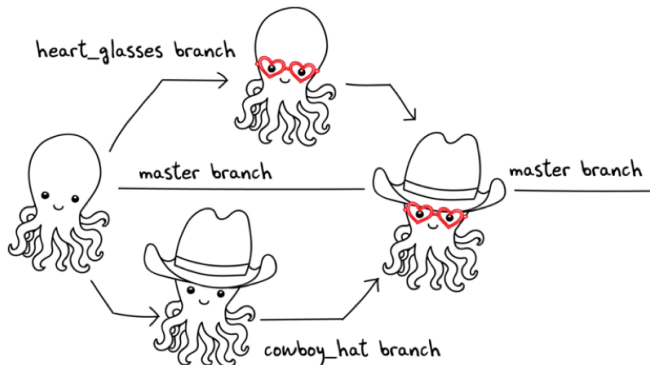
- ▶ `git push`

Cette commande met à jour le dépôt distant avec les changements du dépôt local

**Utilisation** : `git push origin master`

## Les branches Git

- ▶ Une branche dans Git est simplement un pointeur léger et déplaçable vers un des commits
- ▶ La branche par défaut dans Git s'appelle `master`
- ▶ Les branches permettent un développement en parallèle, sans s'affecter entre elles



## Quelques commandes utiles pour travailler avec les branches

► `git branch`

Pour lister toutes les branches locales dans le dépôt actuel et pour créer une nouvelle branche

**Utilisation :** `git branch`

`git branch feature` (*créé une nouvelle  
branche appelée feature*)

`git branch -d feature` (*pour supprimer  
la branche*)

## Quelques commandes utiles pour travailler avec les branches

### ▶ `git branch`

Pour lister toutes les branches locales dans le dépôt actuel et pour créer une nouvelle branche

**Utilisation :** `git branch`

`git branch feature` (*crée une nouvelle  
branche appelée feature*)

`git branch -d feature` (*pour supprimer  
la branche*)

### ▶ `git checkout`

Cette commande est utilisée pour passer d'une branche à l'autre.

**Utilisation :** `git checkout [nom de la branche]`

`git checkout -b [nom de la branche]`

(*pour créer une nouvelle branche et y passe en même temps*)

# Quelques commandes utiles pour travailler avec les branches

- ▶ `git merge`

Cette commande intègre l'historique de la branche spécifiée dans la branche actuelle

**Utilisation** : `git merge [nom de la branche]`



## Quelques commandes utiles pour travailler avec les branches

- ▶ `git merge`

Cette commande intègre l'historique de la branche spécifiée dans la branche actuelle

**Utilisation** : `git merge [nom de la branche]`

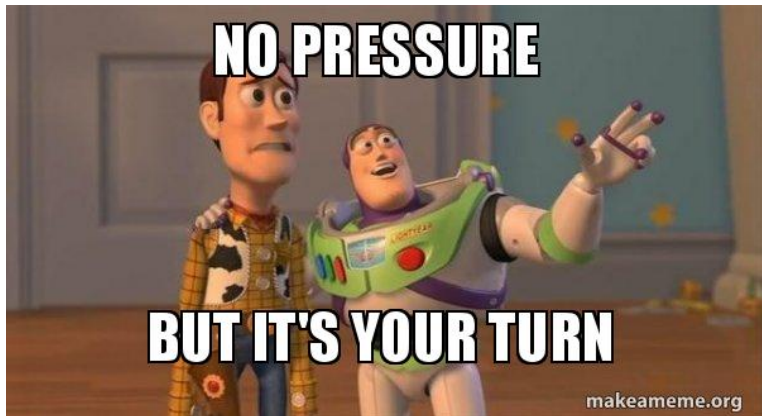
- ▶ `git diff`

Cette commande montre les différences entre les fichiers qui ne sont pas encore dans Index

**Utilisation** : `git diff`

`git diff [première branche] [seconde branche]`  
*(pour voir les différences entre les deux branches)*

A vous de jouer !

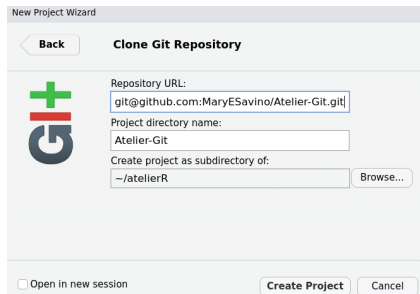
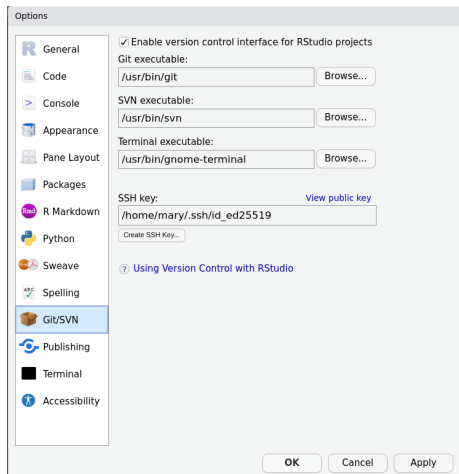


[Lien vers le tutoriel](#)

## Mise en pratique de Git sur Rstudio

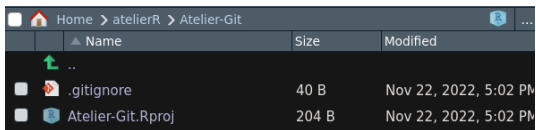
1. Créer un dépôt distant sur Github
2. Le connecter à Rstudio avec clé SSH
3. Créer un fichier avec code
4. Commit, push, pull
5. Créer une nouvelle branche, commit le script sur nouvelle branche
6. Modifier le script sur la nouvelle branche SANS CONFLIT
7. Savoir merge deux branches

# Sur RStudio



Création d'un nouveau projet

## Sur RStudio (2)

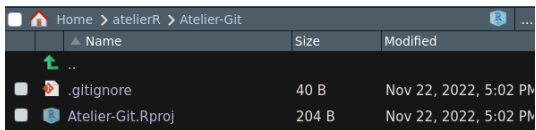


Nouveau projet



Terminal Rstudio

## Sur RStudio (2)



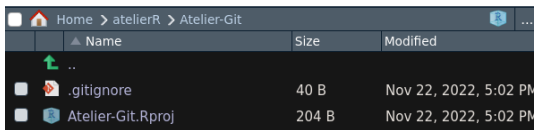
Nouveau projet



Terminal Rstudio

```
git remote add origin adress-https-ou-ssh-du-depot
git push -u origin master
```

## Sur RStudio (2)

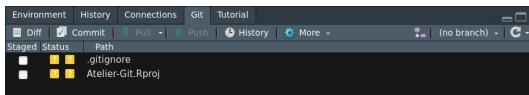


Nouveau projet



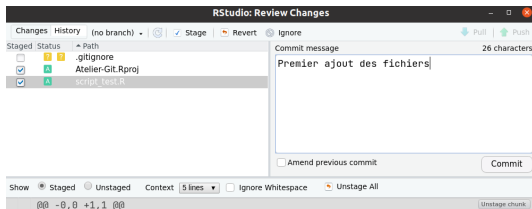
Terminal Rstudio

```
git remote add origin adress-https-ou-ssh-du-depot
git push -u origin master
```

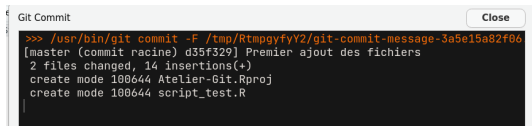


Statut des nouveaux fichiers

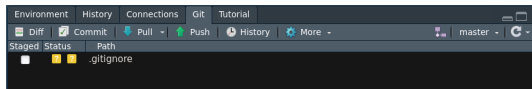
# Sur RStudio (3)



Nouveau projet



Terminal Rstudio



Statut des nouveaux fichiers



## Mise en pratique de Git sur Rstudio

1. Créer un dépôt distant sur Github
2. Le connecter à Rstudio avec clé SSH
3. Créer un fichier avec code
4. Commit, push, pull
5. Créer une nouvelle branche, commit le script sur nouvelle branche
6. Modifier le script sur la nouvelle branche SANS CONFLIT
7. Savoir merge deux branches

## Quelques commandes utiles pour la résolution des conflits

- ▶ Pour remplacer les modifications dans le dépôt de travail par le dernier contenu dans HEAD

```
git checkout -- [nom du fichier]
```

## Quelques commandes utiles pour la résolution des conflits

- ▶ Pour remplacer les modifications dans le dépôt de travail par le dernier contenu dans HEAD

```
git checkout -- [nom du fichier]
```

- ▶ `git reset`

Pour conserver un commit et, par conséquent, supprimer tout ce qui le suit

```
git reset [id du commit]
```

Pour supprimer toutes les modifications et commits locaux, récupérez l'historique le plus récent du serveur distant et faire pointer la branche `master` locale vers lui

```
git fetch origin  
git reset --hard origin/master
```

## Quelques commandes utiles pour la résolution des conflits

- ▶ Pour remplacer les modifications dans le dépôt de travail par le dernier contenu dans HEAD

```
git checkout -- [nom du fichier]
```

- ▶ `git reset`

Pour conserver un commit et, par conséquent, supprimer tout ce qui le suit

```
git reset [id du commit]
```

Pour supprimer toutes les modifications et commits locaux, récupérez l'historique le plus récent du serveur distant et faire pointer la branche `master` locale vers lui

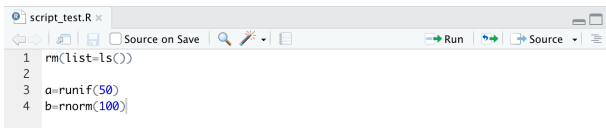
```
git fetch origin  
git reset --hard origin/master
```

- ▶ `git revert`

Pour supprimer un commit

```
git revert [id du commit]
```

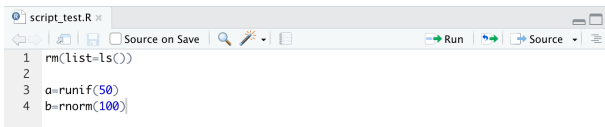
# Sur RStudio



The image shows a screenshot of the RStudio editor window. The title bar indicates the file is named 'script\_test.R'. The toolbar includes icons for navigation, saving, and running code. The source editor contains the following R code:

```
1 rm(list=ls())  
2  
3 a=runif(50)  
4 b=rnorm(100)
```

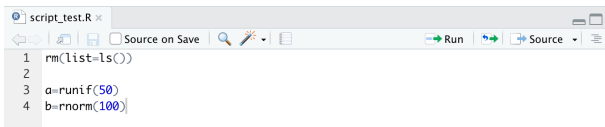
# Sur RStudio



```
script_test.R x
Source on Save
Run
Source
1 rm(list=ls())
2
3 a=runif(50)
4 b=rnorm(100)
```

```
git merge nouvelle_branche
```

# Sur RStudio



```
script_test.R x
Source on Save
Run
Source
1 rm(list=ls())
2
3 a=runif(50)
4 b=rnorm(100)
```

git merge nouvelle\_branche



```
Console Terminal x Background Jobs x
Terminal 1 | ~/Desktop/Atelier-Git
marina:Atelier-Git marina$ git merge nouvelle_branche
Auto-merging script_test.R
CONFLICT (content): Merge conflict in script_test.R
Automatic merge failed; fix conflicts and then commit the result.
```

