

## 3 - Utilisation de Git

### Séries Temporelles avec R - Initiation

Anna Smyk, Tanguy Barthelemy

Insee - Département des Méthodes Statistiques



## Section 1

# Git, GitLab, GitHub... Qu'est-ce-que c'est ?

# Git, GitLab, GitHub... Qu'est-ce-que c'est ?

Avant de rentrer dans les détails d'installation, de configuration ou de ligne de code, expliquons ce que sont ces différents outils.

# Situations initiales

Souvent lorsque vous travaillez sur un projet, vous pouvez vous retrouver dans l'une de ces situations suivantes :

- Je travaille sur un code `extttscript.R`, je le modifie, je crée une nouvelle version `extttscript2.R` ... et je me retrouve avec 15 versions d'un même programme !
- Je travaille avec d'autres personnes sur un projet mais il est difficile d'avancer à plusieurs sur les mêmes fichiers en même temps ? Espace partagés ? Chacun travaille uniquement sur des parties indépendantes ?

# But de Git

**Git** permet de proposer au développeur un versionnage de son code c'est-à-dire :

- créer des versions temporelles de chaque script à chaque modification
- indiquer par un message clair les modifications qui ont été faites
- pouvoir revenir à n'importe quelle version antérieure
- archiver proprement les codes non utilisés
- nettoyer et avoir un répertoire de projet le plus à jour et utile possible
- proposer d'ajouter de la reproductibilité au projet

## Section 2

# GitHub, GitLab

# GitHub, GitLab

Une fois le code bien organisé, versionné, il est intéressant de vouloir le partager avec d'autres personnes.

- pour distribuer le code
- pour travailler à plusieurs

**GitHub** et **GitLab** sont des services de stockage de répertoire. Ils permettent aux développeurs de déposer leur code et de le rendre accessible (lecture et écriture) à tous ou à des personnes autorisées uniquement.

Ainsi une fois le système mis en place, il devient possible de travailler en local et de collaborer à plusieurs via les plateformes GitHub et GitLab.

## gitlab.insee.fr et git.lab.sspcloud.fr

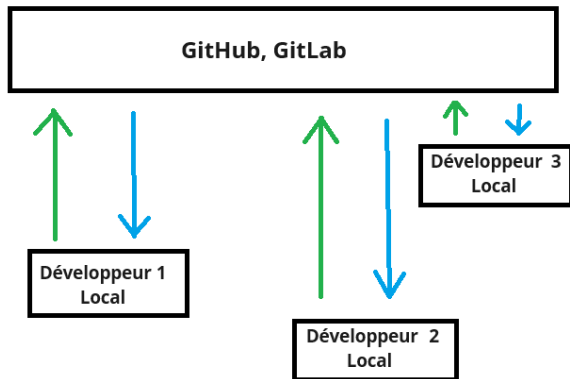
L'Insee possède des espaces GitLab **privés** internes à tous les agents de l'Insee :  
gitlab.insee.fr

Depuis peu, l'Insee possède également via le **datalab** des serveurs GitLab à l'adresse  
git.lab.sspcloud.fr

Ces 2 espaces de travail Git ont pour but de partager et versionner les projets Insee  
(dans un espace privé).



# Workflow



## Section 3

# Git, comment ça marche ?

# Git, comment ça marche ?

Git fonctionne en **ligne de code dans un terminal**.

Mais pas de panique ! Il est possible d'utiliser Git sans avoir à n'écrire une seule ligne de code. RStudio propose une interface pour Git en clique-bouton.

Dans la suite nous allons voir les principales étapes d'utilisation de Git.

# Représentation

On peut se représenter Git sous la forme de photo. Nous prenons des photos d'objets.

Parfois certains objets apparaissent, disparaissent, changent ou ne changent pas.

Lorsqu'on utilise Git, nous choisissons :

- Quand prendre les photos
- Quoi mettre sur les photos
- Comment nommer les photos

Notre projet contient au moins la dernière photo enregistrée par Git mais peut-être plus en avance.

## Etape 0 : git init

Pour initialiser un répertoire Git, il faut toujours commencer par un `git init` dans notre projet.

Nous y reviendrons un peu plus tard.

## Etape 1 : git add

### *Quoi mettre sur la photo ?*

Avec `git add` nous choisissons les fichiers que l'on veut mettre à jour dans notre répertoire.

En reprenant l'analogie de la photo, c'est dans cette étape que je choisis ce que je mets sur la photo.

#### **i** Options

Il est possible de sélectionner  
`git init`  
dans un terminal à l'intérieur du dossier du projet.

[Lien pour la documentation](#)

## Etape 2 : git commit

Ici un **commit** est une photo de notre projet.

La commande :

```
git commit -m "fix arg data.frame to tibble"
```

permet de créer un commit avec comme information principale *"fix arg data.frame to tibble"*.

Il est important d'enregistrer fréquemment son travail et de mettre à jour fréquemment le répertoire Git. Plus il y a de commit, plus on peut retrouver avec précision une erreur ou une modification dans un fichier donné.

[Lien pour la documentation](#)

## Etape online : git push et git pull

Pour cette étape il faut s'assurer que le répertoire online est bien configuré (on en parlera juste après).

Ici on cherche soit à *pousser* (**push**) son travail vers le répertoire distant soit à *tirer* (**pull**) le répertoire.

- [Lien pour la documentation de pull](#)
- [Lien pour la documentation de push](#)



## Section 4

# En production courante

# En production courante

Et c'est tout ! Il suffit de ces étapes pour manipuler Git !

Au quotidien, on enchaîne plutôt les étapes dans l'ordre :

- Avant de commencer à travailler : `git pull` (pour récupérer la dernière version à jour)
- Je travaille et je commite régulièrement mes modifications : `git add file1.R file2.R` et `git commit -m "ajout fonction print"`
- A la fin de mon travail, je veux mettre à jour le répertoire en ligne :
  - Je mets à jour mon projet à jour avec le remote (si quelqu'un d'autre a travaillé en même temps que moi sur les mêmes fichiers) : `git pull`
  - Enfin je peux *pusher* mes modifications : `git push`

## Vocabulaire :

Le vocabulaire `init`, `add`, `commit` se rapporte au répertoire en local. Par opposition, ce qui se rapporte au online est appelé **remote** (à distance) : `pull`, `push` et `clone`.

L'opération `git clone` permet de récupérer un répertoire Git qui se trouve en ligne.

## Section 5

# Installation et configuration

# Installation de Git

Git est téléchargeable [ici](#).

Git est déjà installé sur les sessions AUS et SSPCloud.

Il est possible (et recommandé) d'installer Git sur son poste.

# Configuration Git

Pour commencer, il faut récupérer un répertoire avec Git.

On se place ici dans le cadre d'un projet R développé avec RStudio.

3 scénarios se présentent :

- Un nouveau projet vide
- Un projet existant sans Git
- Un projet distant avec Git

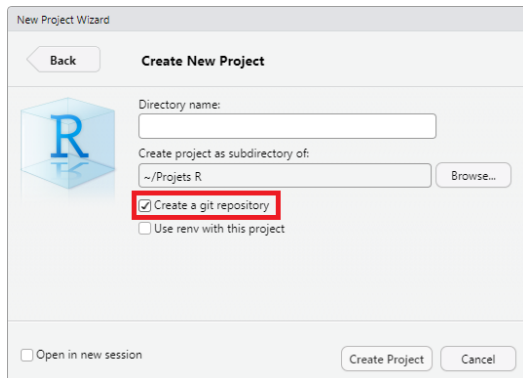
## Terminal

Les 2 premiers cas reviennent à lancer

```
git init
```

dans un terminal à l'intérieur du dossier du projet.

# Un nouveau projet vide



New Project Wizard

Back Create New Project

Directory name:

Create project as subdirectory of:

~/Projets R Browse...

☒ Create a git repository

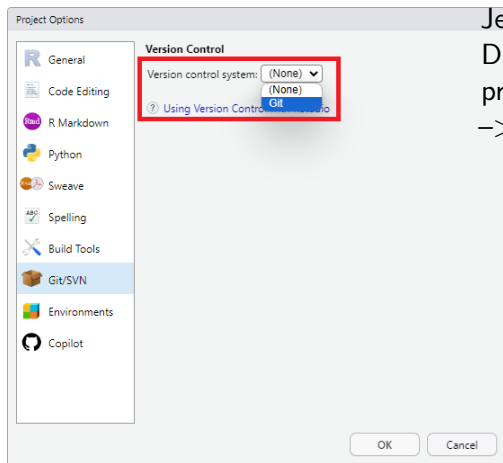
☐ Use renv with this project

☐ Open in new session

Create Project Cancel

Je commence un nouveau projet de 0. Lors de la création de mon projet avec RStudio, je choisis l'option : ***Create a git repository***

# Un projet existant sans Git



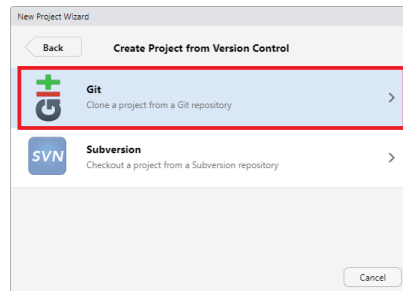
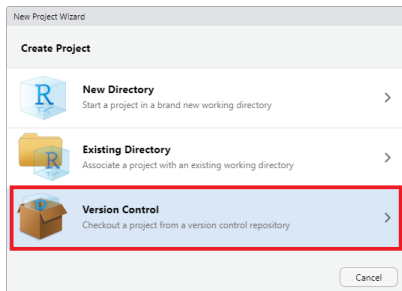
Je souhaite utiliser Git sur un projet déjà existant. Dans ce cas, là il faut aller dans les options du projet (**Tools** → **Project Options** → **Git/SVN** → **Version control system**) et choisir **Git**



# Un projet distant avec Git

Dans ce cas, on utilisera `git clone` pour cloner un répertoire online en local.

Sous RStudio, lors de la création de projet, on choisit l'option **Version Control** → **Git**. Puis on précise l'adresse URL du répertoire.



# Configuration de GitHub

Ici nous prenons l'exemple de GitHub.

- ① Création de compte : il faut se créer un compte GitHub.
- ② Créer un répertoire qui contiendra votre projet.
- ③ Créer un token avec les droits en lecture et écriture sur l'ensemble des répertoires.
- ④ Lier le répertoire local au répertoire distant.

Pour la dernière étape de configuration, il faut s'appuyer sur les indications fournies par GitHub.

# Remote

Détail de certaines instruction que l'on peut trouver :

```
git remote add origin https://github.com/user/project-name.git
```

Cette instruction sert à dire au projet git : “On ajoute un répertoire remote qui s'appellera origin et qui se trouve à l'adresse <https://github.com/user/project-name.git>”

# Branche

```
git push -u origin main
```

Ici on essaye de mettre à jour le répertoire remote origin avec nos modifications locales qui se trouvent sur la branche main.

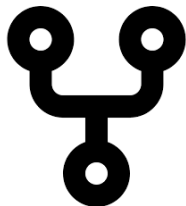
La commande `-u` ou `--set-upstream` permet d'indiquer le remote reference à git pour les futurs push.

Le sujet des branches sur Git est un sujet important car il permet une seconde organisation du travail (parallèle aux versions linéaires d'un projet).

C'est un sujet inévitable pour la gestion de conflit !

Mais nous n'en parlerons pas beaucoup plus.

# Fork



Une fois un répertoire distant créé, il est possible de **forker** le répertoire chez soi (en ligne dans mon espace personnel). Cela a pour conséquence de créer une *copie* du repo avec une relation d'ascendance entre le repertoire d'origine (appelé **upstream**) et le repertoire forké (**origin** actif).

Comme pour les branches, l'intérêt est de pouvoir travailler en parallèle dans le projet principal

# Documentation

La [documentation de Git](#) est très complète, accessible et compréhensible mais en anglais 🇬🇧.

Ne pas hésiter à rechercher par mots-clé.

Aussi GitHub et GitLab regorgent de documentation (en anglais 🇬🇧 et en français 🇫🇷) pour la plupart des manipulations.

# Questions

Avez-vous des questions ?