Capsule 1 – Concepts de base de Git, GitHub et du versionnage

Hubert Cadieux et Arnaud Beaulé

Introduction (50 sec)

Bonjour tout le monde!

Vous êtes nombreux à découvrir la programmation et les outils de gestion de code dans le cadre de vos études et de vos recherches en sciences sociales.

Vous utilisez ces outils pour faire des analyses de données, pour écrire des articles, pour créer des rapports de recherche, ou encore pour gérer vos bases de données.

Mais pour pouvoir collaborer efficacement, gérer les différentes versions de vos fichiers, et assurer la reproductibilité de vos travaux, il faut des outils spécialisés.

Dans cette capsule, nous allons découvrir Git et GitHub, deux outils de *contrôle de version* qui vous permettront de mieux organiser et partager vos projets de recherche.

Définition de Git et GitHub (2 min)

Premièrement, qu'est-ce qu'un système de contrôle de version ?

Imaginez que vous travaillez sur votre mémoire de maîtrise. Vous travaillez dessus, vous l'envoyez à votre directrice pour de la rétroaction, elle vous renvoie ses commentaires, vous les appliquez et ainsi de suite.

Au bout d'un moment, vous vous retrouvez avec des fichiers qui s'appellent `memoire_final.docx`, `memoire_final_vraiment_final.docx`, `memoire_final_pour_vrai_cette_fois.docx`. Ça vous dit quelque chose ?

Un système de contrôle de version, c'est exactement l'outil qui résout ce problème. Il garde automatiquement un historique de toutes vos modifications, vous permet de revenir à n'importe

quelle version précédente, et surtout, il vous permet de voir exactement ce qui a changé entre deux versions. C'est exactement à ça que Git et GitHub servent.

Maintenant, la différence entre Git et GitHub. C'est important de bien comprendre ça.

Git, c'est le logiciel que vous installez sur votre ordinateur. C'est lui qui fait tout le travail de gestion des versions, qui suit vos modifications, qui crée l'historique. Git fonctionne entièrement en local sur votre machine.

GitHub, c'est une plateforme web qui utilise Git. C'est comme un Google Drive, mais spécialisé pour les projets de programmation et de recherche. GitHub vous permet de sauvegarder vos projets en ligne, de les partager avec d'autres personnes et de collaborer facilement, même à distance.

Ce qui rend GitHub particulièrement intéressant, c'est que son interface est conçue pour être accessible aux débutants. Sans GitHub, utiliser Git directement nécessiterait beaucoup plus de connaissances techniques et l'utilisation constante de la ligne de commande. GitHub rend donc l'utilisation de Git beaucoup plus facile et visuelle.

Si ça semble abstrait, ne vous inquiétez pas. Plus tard dans ces capsules, nous allons vous montrer concrètement ces différences pour que vous compreniez bien comment tout ça fonctionne en pratique.

Pourquoi utiliser Git/GitHub en sciences sociales ? (1 min 20 sec)

Maintenant, vous vous demandez probablement : "Pourquoi est-ce que moi, étudiant en sciences sociales, j'aurais besoin de ça ?" C'est une excellente question !

Premièrement, la traçabilité des modifications. Quand vous travaillez sur une analyse de données ou un projet de recherche, vous voulez pouvoir documenter exactement ce que vous avez fait, quand vous l'avez fait, et pourquoi. Git garde une trace complète de tous vos changements avec vos commentaires. Plus jamais de "je ne me rappelle plus pourquoi j'ai supprimé cette partie de l'analyse".

Deuxièmement, la collaboration et le partage. En sciences sociales, vous travaillez souvent en équipe - avec votre directeur ou directrice de recherche, des collègues étudiants ou dans des projets interdisciplinaires. GitHub vous permet de travailler ensemble sur les mêmes fichiers sans vous marcher sur les pieds, de partager facilement vos données et vos codes d'analyse.

Finalement, la reproductibilité. C'est un enjeu majeur en recherche aujourd'hui. Avec Git et GitHub, quelqu'un d'autre peut reprendre exactement votre travail, comprendre vos étapes, et reproduire vos résultats. C'est essentiel pour la crédibilité scientifique.

Vocabulaire de base (1 min 30)

Maintenant, tournons-nous au vocabulaire de base dont vous vous servirez lors de vos utilisations de GitHub. Pour utiliser Git et GitHub efficacement, il est essentiel de bien comprendre quelques termes clés. Ces notions sont le fondement même de la collaboration sur un projet digital : elles permettent de s'organiser, de suivre les évolutions de son travail, et de travailler à plusieurs sans confusion.

Dépôt (repository/repo)

Qu'est-ce que c'est ? Le dépôt est le centre du projet. Il contient tous les fichiers, leur historique de modifications et toutes les informations sur les contributions des membres de l'équipe.

Pourquoi est-ce important ? C'est le point de départ de tout projet collaboratif. Sans dépôt, il n'y a pas de versionnage ni de suivi des changements: chaque membre travaillerait dans son coin sans partager son travail.

Commit (validation/livraison)

Qu'est-ce que c'est ? Un commit est l'action d'enregistrer une série de modifications dans le dépôt, accompagnée d'un message qui explique ce qui a été fait.

Pourquoi est-ce important ? Les commits permettent de garder une trace précise de chaque étape du projet. Si une erreur survient, on peut revenir en arrière ou comprendre qui a fait quoi, et pourquoi.

Branche (branch)

Qu'est-ce que c'est ? Une branche est une version parallèle du projet. Elle permet de travailler sur une nouvelle idée ou correction sans modifier le travail principal.

Pourquoi est-ce important ? Les branches évitent les conflits entre les membres de l'équipe et permettent d'expérimenter sans risque. On peut développer de nouvelles fonctionnalités ou corriger des bugs sans toucher au reste du projet.

Fusion (merge)

Qu'est-ce que c'est ? La fusion consiste à intégrer les modifications d'une branche dans une autre, par exemple pour ajouter une nouvelle fonctionnalité au projet principal.

Pourquoi est-ce important ? C'est grâce à la fusion que le travail de chacun devient partie intégrante du projet. Sans fusion, chaque branche resterait isolée et le projet ne progresserait pas.

Demande de fusion (pull request)

Qu'est-ce que c'est ? Une demande de fusion permet de proposer des modifications à intégrer dans le projet principal. Les autres membres de l'équipe peuvent examiner, discuter et valider ces changements avant qu'ils ne soient effectivement fusionnés.

Pourquoi est-ce important ? Les demandes de fusion favorisent la collaboration et la qualité du code. Elles permettent de discuter des améliorations, de corriger les erreurs et d'apprendre des autres.

Bien comprendre ce vocabulaire est crucial pour collaborer efficacement sur un projet numérique. Ces outils vous aideront à organiser votre travail, à garder une trace de chaque étape, à travailler à plusieurs sur les mêmes fichiers, et à garantir la qualité et la cohérence de votre projet. Dans la prochaine section, nous verrons comment installer Git et utiliser GitHub pour commencer à travailler concrètement.

Installation de Git (2 min)

Avant de pouvoir collaborer sur un projet avec Git et GitHub, il est nécessaire d'installer Git sur votre ordinateur. Heureusement, l'installation est simple et rapide, que vous utilisiez Windows, Mac ou Linux.

Installation sur Windows, Mac et Linux

Windows : Téléchargez l'installateur officiel depuis le site web de Git, puis suivez les instructions du programme d'installation. Après l'installation, ouvrez l'invite de commandes ou Git Bash et tapez git –version pour vérifier que tout fonctionne.

Mac : Ouvrez le Terminal et tapez git –version. Si Git n'est pas installé, le système vous proposera de l'installer via les Xcode Command Line Tools. Vous pouvez aussi utiliser l'installateur officiel ou Homebrew.

Linux : Utilisez le gestionnaire de paquets de votre distribution (par exemple, sudo apt install git sur Ubuntu ou sudo dnf install git sur Fedora).

Introduction à GitHub Desktop (GUI) et au terminal (CLI)

Terminal (CLI) : La méthode traditionnelle pour utiliser Git se fait via la ligne de commande. Elle offre un contrôle précis sur toutes les actions et est disponible sur tous les systèmes d'exploitation.

GitHub Desktop (GUI) : Pour ceux qui préfèrent une interface graphique, GitHub Desktop permet de gérer les dépôts et les commandes Git de façon visuelle, sans avoir à taper de commandes. Il installe automatiquement Git si besoin.

Avantages et limites de chaque approche

-**Terminal (CLI)** :

-*Avantages* :GitHub offre une flexibilité totale ainsi qu'un accès à toutes les fonctionnalités de Git. Il est aussi possible de créer des scripts automatisables, et donc de sauver beaucoup de temps!

-*Limites* : Courbe d'apprentissage plus raide pour les débutants, nécessite de connaître les commandes.

-**GitHub Desktop (GUI)** :

-*Avantages* : C'est une interface intuitive, adaptée aux débutants, offrant des visualisations claires des modifications et des branches.

-*Limites* : Moins de contrôle sur les fonctionnalités avancées, dépendance à l'interface graphique de l'outil.

Avec Git installé, vous êtes prêt à collaborer sur n'importe quel projet!

Présentation de l'interface GitHub (2 min)

Pour profiter pleinement de la collaboration sur GitHub, il faut d'abord comprendre comment naviguer dans l'interface et utiliser ses principales fonctionnalités.

Création de compte

Inscription : Rendez-vous sur github.com et cliquez sur «Sign up». Suivez les instructions pour créer un compte: choisissez un nom d'utilisateur, une adresse e-mail et un mot de passe sécurisé.

Vérification : Après l'inscription, vérifiez votre adresse e-mail pour activer votre compte. Vous pouvez ensuite personnaliser votre profil et découvrir les projets publics.

Navigation dans un dépôt (repo)

Page principale : Lorsque vous accédez à un dépôt (repository), la première page affiche généralement le fichier *README*, qui explique le projet, ses objectifs et comment y contribuer.

Exploration des fichiers : L'onglet «Code» permet de parcourir tous les fichiers et dossiers du projet. Vous pouvez visualiser, modifier ou télécharger les fichiers directement depuis l'interface web.

Historique des modifications : L'onglet «Commits» montre l'historique de toutes les modifications apportées au projet, avec les messages associés à chaque commit. Cela permet de suivre l'évolution du projet et de comprendre qui a fait quoi.