Usando Git y GitHub desde RStudio: : CHEATSHEET

El **control de versiones**, también conocido como **control del código fuente**, es la práctica de rastrear y gestionar los cambios en el código del software.

Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo.

Git es un software de **código abierto** para el control de versiones, desarrollado originalmente en 2005 por Linus Torvalds, el creador del núcleo del sistema operativo Linux.

Git es una herramienta de control de versiones para seguir los cambios en el código fuente de un proyecto.

Github es el servicio de alojamiento más popular para colaborar en el código utilizando Git.

Requisitos

1. R y RStudio instalados
2. Git instalado
3. Cuenta gratuita en Github

Revisar que Git está instalado

En la terminal de RStudio, tipear `which git` para solicitar la ruta de tu Git ejecutable:

```
which git
## /usr/bin/git
```

y `git --version` para ver su versión:

```
git --version
## git version 2.34.1
```

Presentarse a Git

Abrir un shell en RStudio (*Tools > Shell*) y tipear cada línea por separado sustituyendo su nombre y el correo electrónico asociado con su cuenta de Github:

```
git config --global user.name 'Jane Doe'
git config --global user.email 'jane@example.com'
```

Glosario de Github

Este glosario presenta la terminología común de Git y GitHub.

Comandos básicos

`git init <directorio>`	Crear un repositorio vacío en Git en un directorio específico.
`git clone <repositorio>`	Clonar un repositorio ubicado en <repositorio> en tu dispositivo local.
`git config user.name <usuario>`	Definir el nombre del autor para ser usado en todos los commits en el repositorio local.
`git add <directorio>`	Agregar todos los cambios en <directorio> para el siguiente commit.
`git commit -m <"mensaje">`	Hacer *commit* a los cambios, pero en lugar de abrir un editor de texto, usar <"mensaje"> como mensaje de confirmación.
`git status`	Enlistar los archivos que están dentro o fuera del área de ensayo (o *index*) y los que no están siendo rastreados.
`git log`	Mostrar el historial de commits usando el formato por defecto.
`git diff`	Mostrar las diferencias entre el *index* y el directorio de trabajo.

Repositorios remotos

`git remote add <nombre> <url>`	Crear una nueva conección con un repositorio remoto. Una vez creada, se puede usar <nombre> como un atajo para <url> en otros comandos.
`git fetch <remoto> <branch>`	Recuperar un <branch> específico de <remoto>, desde el repositorio local. Si no se incluye <branch>, se recuperarán todas las referencias.
`git pull <remoto>`	Recuperar la copia del origen remoto especificado de la rama actual e **inmediatamente** fusionarla en el repositorio local.
`git push <remoto> <branch>`	Sube una rama a <remote>, junto con los commits y objetos necesarios. Además, si no existe, crea una rama nombrada en el repositorio remoto.

Deshacer cambios

`git revert <commit>`	Crear un nuevo *commit* que deshaga todos los cambios hechos en <commit> y aplicarlo en la rama actual.
`git reset <archivo>`	Eliminar <archivo> del *index* dejando el directorio de trabajo sin modificar. Esto desestructura el archivo sin sobrescribir los cambios.
`git clean -n`	Mostrar los archivos que se eliminarán del directorio de trabajo. Usar –f en lugar de –n para ejecutar esta eliminación.

Reescribir el historial de Git

`git commit --amend`	Reemplazar el último *commit* combinando los cambios preparados con el *commit* anterior en lugar de crear uno nuevo. Usarlo cuando no hay nada preparado permite editar el mensaje del último *commit*.
`git rebase <base>`	Cambiar la base de la rama actual a <base>, la cual puede ser un ID, un nombre de rama, una etiqueta o una referencia relativa a HEAD.
`git reflog`	Muestra un registro de cambios en el HEAD del repositorio local. Añadir --relative-date para mostrar la información de la fecha o --all para mostrar todas las referencias.

Ramas en Git

`git branch`	Enlistar todas las ramas del repositorio local. Añadir el argumento <branch> para crear una nueva rama con el nombre <branch>.
`git checkout –b <branch>`	Crear y extraer una nueva rama con el nombre <branch> simultáneamente. Retire el –b para ejecutar «git branch».
`git merge <branch>`	Combinar <branch> con la rama actual.