Using Git and GitHub with RStudio: : CHEATSHEET

Version control control, also known as **source control**, is the practice of tracking and managing changes to software code.

Version control systems are software tools that help software teams manage changes to source code over time.

Git is an **open-source** software for version control, originally developed in 2005 by Linus Torvalds, the creator of the Linux operating system kernel.

Git it is a version control tool to track the changes in the source code of a project.

Github is the most popular hosting service for collaborating on code using Git.

Requirements

1. R and RStudio installed
2. Git installed
3. Register a free Github account

Check that Git is installed

In the Terminal of RStudio, enter which git to request the path to your Git executable:

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

and git --version to see its version:

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

Introduce yourself to Git

Open a shell from RStudio *Tools* > *Shell* and type each line separately by substituting your name and the email associated with your GitHub account:

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

Github Glossary

This glossary introduces common Git and GitHub terminology.

Basics

git init <directory>	Create empty Git repository in specified directory.
git clone <repository>	Clone a repository located at <repository> on your local machine.
git config user.name <username>	Define author name to be used for all commits in current repository.
git add <directory>	Stage all changes in <directory> for the next commit.
git commit -m <"message">	Commit the staged snapshot, but instead of launching a text editor, use <"message"> as the commit message.
git status	List which files are staged, unstaged, and untracked.
git log	Display the entire commit history using the default format.
git diff	Show unstaged changes between your index and working directory.

Remote Repositories

git remote add <name> <url>	Create a new connection to a remote repository. After adding a remote, you can use <name> as a shortcut for <url> in other commands.
git fetch <remote> <branch>	Fetches a specific <branch>, from the repository. Leave off <branch> to fetch all remote refs.
git pull <remote>	Fetch the specified remote's copy of current branch and **immediately** merge it into the local copy.
git push <remote> <branch>	Push the branch to <remote>, along with necessary commits and objects. Creates named branch in the remote repository if it doesn't exist.

Undoing Changes

git revert <commit>	Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch.
git reset <file>	Remove <file> from the staging area but leave the working directory unchanged. This unstages a file without overwriting any changes.
git clean -n	Shows which files would be removed from working directory. Use the –f flag in place of the –n flag to execute the clean.

Rewriting Git History

git commit --amend	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
git rebase <base>	Rebase the current branch onto <base>. <base> can be a commit ID, branch name, a tag, or a relative reference to HEAD.
git reflog	Show a log of changes to the local repository's HEAD. Add --relative-date flag to show date info or --all to show all refs.

Git Branches

git branch	List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>.
git checkout –b <branch>	Create and check out a new named <branch>. Drop the –b flag to checkout an existing branch.
git merge <branch>	Merge <branch> into the current branch.