GEO 242: Numerical methods and modeling in the geosciences

git and versioning

Outline

Why version your codes?

git and github

Code versioning

Versioning is the process of assigning unique identifiers (names or numbers) to each version of computer codes or software (or indeed other entities like text documents)

Having a record of changes to the code allows you to isolate potential bugs introduced by those changes, and for end users to check whether the code they are using is up to date.

It also promotes reproducibility, by allowing users to specify exactly which version of a code they used to produce their results.

Versioning schemes

Most versioning schemes are numerical, with higher numbers indicating more recent versions, e.g. Microsoft Windows (currently version 11)

Some codes use date-based or 'calendar' versioning, e.g. Ubuntu, currently in version 25.10 (October 2025)

For greater granularity, however, best practice has solidified around more hierarchical versioning schemes…

Semantic versioning

Probably the most used versioning scheme at present is 'semantic versioning'

Versioning is described by a three-part version number:

Major.Minor.Patch (e.g. 1.2.42)

Some developers add an optional 'pre-release tag' (e.g. -alpha, -beta, -rc) to the end to provide a sense of stability/confidence to users

Semantic versioning

The significance of each number in the hierarchy is up to the developer, but general rules of thumb are in existence

Major versions may imply additions of multiple new features, or changes in compatibility (files used in v1.0.0 may not work in v2.0.0, say)

Minor versions may imply significant additions (a new feature, a new efficiency improvement)

Patches imply changes to one aspect of a code (bug fixes, say)

git

The most commonly used versioning software in current use is called 'git'

git was originally developed by Linus Torvalds, the developer of Linux, in 2005 as a means of tracking changes to the code base for the Linux kernel. It is now managed by a dedicated team

git is free and open source

git

From the readme file:

"git" can mean anything, depending on your mood.

Random three-letter combination that is pronounceable, and not actually used by any common UNIX command. The fact that it is a mispronunciation of "get" may or may not be relevant.

Stupid. Contemptible and despicable. Simple. Take your pick from the dictionary of slang.

"Global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room.

"Goddamn idiotic truckload of sh*t": when it breaks.

git

The main principles of git are:

Changes are tracked locally

- This makes tracking changes more efficient

It allows nonlinear development

- Alternate versions, 'branches' and 'forks' can be established easily, and later merged back into the main version

It is compatible with Internet protocols

- Local code versions can be easily synced to remote repositories

GitHub and open source

A major benefit of using git is that it connects seamlessly with online repositories, like GitHub

GitHub is a free website that hosts users' code repositories, and facilitates sharing of codes and open source development

Users can 'clone' these remote repositories, modify them on their own machines, then 'push' changes back to the repository

Cloning

```
git clone <url>
```

This will clone the repository at the given URL to your current directory, and establish a git database there that includes the version history

Updating your cloned version

```
git pull
```

This will check the online repository for changes, and update the local versions of the files with the new versions

Making changes and updating

```
git add <filename>
```

This will add the named file to the 'staging area' – the files that are being prepared to be pushed to the remote repository

```
git rm <filename>
```

This will mark a file for deletion on the remote repository and delete it locally. Be careful!

Making changes and updating

```
git commit -m "[descriptive message]"
```

This tags your proposed changes with a descriptive message, that hopefully describes what was changed. This is a mandatory step before pushing the changes to the remote repository.

```
git push
```

This pushes your changes back to the repository (this may require inputting setting up a 'token' that is tied to your account, and also setting up a ‘key’ on your local machine)

Branches and forks

You do not have to make edits directly to the original versions of the files (the "main branch") of your codes – this may not be desirable

Instead, you can make a new version of your code, a named "branch" that you can change, and once you are happy with those changes, and push them to the repository, you can merge them into the main branch

Branches and forks

Forks are a similar concept, but are typically used to make your own copy of somebody else's repository

You may not have permission to push your own versions of fixes to the original repository, but you can do it with your own copy

You can then issue a 'pull request' to the original repository to ask them to incorporate the changes in your fork into the main repository. Usually the original developer will review these changes and accept them, or not, as the case may be…