Using Git for Comp-Neuro

Seb James

ABRG Sheffield Internal Seminar

2018/06/28

Introduction

- This seminar is about a command-line tool called Git.
- There's going to be a lot of text in these slides (sorry!)
- I'll give an overview of Git and why it's helpful to use it, then we'll go through some example tasks together.

What is Git?

Git is a **Revision Control** or **Version Control** tool.
Revision control has two main features:

What is Git?

Git is a **Revision Control** or **Version Control** tool.
Revision control has two main features:

1. Revision control allows you to keep references to different versions of a single file

What is Git?

Git is a **Revision Control** or **Version Control** tool.
Revision control has two main features:

1. Revision control allows you to keep references to different versions of a single file without having to explicitly make copies

What is Git?

Git is a **Revision Control** or **Version Control** tool.
Revision control has two main features:

1. Revision control allows you to keep references to different versions of a single file without having to explicitly make copies
2. Most revision control tools allow several people to work on the same files

File revisions

I bet you have folders that look like this:

File revisions

I bet you have folders that look like this:

- myCoolProgram.cpp

File revisions

I bet you have folders that look like this:

- myCoolProgram.cpp
- myCoolProgram_old.cpp

File revisions

I bet you have folders that look like this:

- myCoolProgram.cpp
- myCoolProgram_old.cpp
- myCoolProgram_1.cpp

File revisions

I bet you have folders that look like this:

- myCoolProgram.cpp
- myCoolProgram_old.cpp
- myCoolProgram_1.cpp
- myCoolProgram_thisOneWorked.cpp
- myCoolProgram_whoKnowsWhatThisOneIs.cpp

File revisions

I bet you have folders that look like this:

- myCoolProgram.cpp
- myCoolProgram_old.cpp
- myCoolProgram_1.cpp
- myCoolProgram_thisOneWorked.cpp
- myCoolProgram_whoKnowsWhatThisOneIs.cpp

With revision control, you only have

- myCoolProgram.cpp

File revisions

I bet you have folders that look like this:

- myCoolProgram.cpp
- myCoolProgram_old.cpp
- myCoolProgram_1.cpp
- myCoolProgram_thisOneWorked.cpp
- myCoolProgram_whoKnowsWhatThisOneIs.cpp

With revision control, you only have

- myCoolProgram.cpp

You might then have branches like **master**, **tryingIdea1**, **PlosCompBio2018** and so on.

Project revisions

The ability to branch and tag *really* comes into its own when you have any kind of project with more than one file in it.
All of a sudden you find yourself needing one folder for each version of your project.

Working with other people

(This is ‘main feature 2’)

- Suppose you have some code which is used by yourself and 5 of your colleagues - some sort of library.

Working with other people

(This is 'main feature 2')

- Suppose you have some code which is used by yourself and 5 of your colleagues - some sort of library.
- You find an error and fix it. Now you have to email the fix to *5* people.

Working with other people

(This is 'main feature 2')

- Suppose you have some code which is used by yourself and 5 of your colleagues - some sort of library.
- You find an error and fix it. Now you have to email the fix to *5* people.
- Just after you emailed them, you find an error in your fix, and you have to send another email...

Working with other people

(This is 'main feature 2')

- Suppose you have some code which is used by yourself and 5 of your colleagues - some sort of library.
- You find an error and fix it. Now you have to email the fix to *5* people.
- Just after you emailed them, you find an error in your fix, and you have to send another email...
- Now imagine that one of your colleagues found a separate fix in the same file and emails that around. Which fix is more important? Who is going to merge the two fixes together?

Working with other people

(This is 'main feature 2')

- Suppose you have some code which is used by yourself and 5 of your colleagues - some sort of library.
- You find an error and fix it. Now you have to email the fix to *5* people.
- Just after you emailed them, you find an error in your fix, and you have to send another email...
- Now imagine that one of your colleagues found a separate fix in the same file and emails that around. Which fix is more important? Who is going to merge the two fixes together?

You need a revision control system to automate the process of distributing changes between users of a set of files, and to manage the merging of changes together.

Other revision control systems

- RCS (Revision Control System)
- SCCS (Source Code Control System)
- CVS (Concurrent Versions System)
- Subversion
- Tons of proprietary systems
- Bazaar
- BitKeeper
- Mercurial

Other revision control systems

- RCS (Revision Control System)
- SCCS (Source Code Control System)
- CVS (Concurrent Versions System)
- Subversion
- Tons of proprietary systems
- Bazaar
- BitKeeper
- Mercurial

Git is not the first revision control system, and its developers could draw on a lot of collective knowledge when designing it.

Why did someone develop Git?

Why did someone develop Git?

- Most revision control tools have been pretty good at feature 1 (file versioning)

Why did someone develop Git?

- Most revision control tools have been pretty good at feature 1 (file versioning)
- ...but not great at managing multiple contributions

Why did someone develop Git?

- Most revision control tools have been pretty good at feature 1 (file versioning)
- ...but not great at managing multiple contributions
- That caused Linus Torvalds to use the proprietary BitKeeper from 2002 to manage the Linux code base.

Why did someone develop Git?

- Most revision control tools have been pretty good at feature 1 (file versioning)
- ...but not great at managing multiple contributions
- That caused Linus Torvalds to use the proprietary BitKeeper from 2002 to manage the Linux code base.
- In 2005 Linus fell out with BitMover Inc., and Git was created to replace it.

Why did someone develop Git?

- Most revision control tools have been pretty good at feature 1 (file versioning)
- ...but not great at managing multiple contributions
- That caused Linus Torvalds to use the proprietary BitKeeper from 2002 to manage the Linux code base.
- In 2005 Linus fell out with BitMover Inc., and Git was created to replace it.
- The name *git* doesn't really mean anything.

What's different about Git?

- Git is a *distributed* revision control system

What's different about Git?

- Git is a *distributed* revision control system
- It doesn't have the classical client-server architecture...

What's different about Git?

- Git is a *distributed* revision control system
- It doesn't have the classical client-server architecture...
- ...although typically you will work with a common remote repository as your upstream source.

What's different about Git?

- Git is a *distributed* revision control system
- It doesn't have the classical client-server architecture...
- ...although typically you will work with a common remote repository as your upstream source.
- When you clone a repository from a source, you have *everything* (all the file history and meta-data) in those files to become a source for someone else.

What's different about Git?

- Git is a *distributed* revision control system
- It doesn't have the classical client-server architecture...
- ...although typically you will work with a common remote repository as your upstream source.
- When you clone a repository from a source, you have *everything* (all the file history and meta-data) in those files to become a source for someone else.
- That means you can work on your code, making incremental commits even when you don't have internet access.

What's different about Git?

- Git is a *distributed* revision control system
- It doesn't have the classical client-server architecture...
- ...although typically you will work with a common remote repository as your upstream source.
- When you clone a repository from a source, you have *everything* (all the file history and meta-data) in those files to become a source for someone else.
- That means you can work on your code, making incremental commits even when you don't have internet access.
- And every copy of the repo is a backup!

Git is *not* github.com

- Github is a commercial website which makes it easy to use Git
- bitbucket.org is an alternative

Git is *not* github.com

- Github is a commercial website which makes it easy to use Git
- bitbucket.org is an alternative
- Generally, public hosting is free, private hosting incurs a fee

Git is *not* github.com

- Github is a commercial website which makes it easy to use Git
- bitbucket.org is an alternative
- Generally, public hosting is free, private hosting incurs a fee
- It's pretty easy to host a git repository yourself, but the nice web interface has made github.com very popular for source code hosting

Git is *not* github.com

- Github is a commercial website which makes it easy to use Git
- bitbucket.org is an alternative
- Generally, public hosting is free, private hosting incurs a fee
- It's pretty easy to host a git repository yourself, but the nice web interface has made github.com very popular for source code hosting
- It's now a big business; it was acquired by Microsoft in early June

Why is Git good for us?

We don't have hundreds of people working on the same files, but...

Why is Git good for us?

We don't have hundreds of people working on the same files, but...

- We write code, so that's very natural to hold in revision control (also XML)

Why is Git good for us?

We don't have hundreds of people working on the same files, but...

- We write code, so that's very natural to hold in revision control (also XML)
- We have a frequent need to tag our work (e.g. to match up with a paper or document)

Why is Git good for us?

We don't have hundreds of people working on the same files, but...

- We write code, so that's very natural to hold in revision control (also XML)
- We have a frequent need to tag our work (e.g. to match up with a paper or document)
- You can include your paper alongside your model code in a single, public repository

Why is Git good for us?

We don't have hundreds of people working on the same files, but...

- We write code, so that's very natural to hold in revision control (also XML)
- We have a frequent need to tag our work (e.g. to match up with a paper or document)
- You can include your paper alongside your model code in a single, public repository
- Use of Github is a very effective way to share your published models with your peers

The Git Tutorial

The rest of these pages have been put together from material taken from the Software Carpentry project, which emphasises the use of Git.
Head over to:
http://sebjameswml.github.io/git-novice/
And start on the lesson “A better kind of Backup”