Version control with Git – an introduction

The work of statistical programmers, statisticians and data scientists today requires more and more knowledge of software development, and especially of version control. [5] The use of R and other open-source software in pharma is increasing, and as more open-source code is being developed, there are big benefits of collaborating with other authors. Git is a version control system that is rapidly becoming the standard for open-source projects. In this article, we'd like to explain what Git is, what its basic features are and why you should start using it.

Git is a free and open-source version control system for tracking changes in source code during software development. Git was created by Linus Torvalds in 2005 for development of the Linux kernel. With his dry wit, he has affectionately called the tool the information manager from hell. [1]

Version control also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information. Version control systems can be roughly divided into two categories: Centralized Version Control System (CVCS) and Distributed Version Control System (DVCS). [2] Git is a Distributed Version Control System.

A DVCS is a form of version control where the complete codebase - including its full history - is mirrored on every developer's computer. It takes a peer-to-peer approach to version control, as opposed to the client–server approach of centralized systems. Distributed revision control synchronizes repositories by exchanging patches from peer to peer.

Some of the basic operations in Git are [2]:

1. Initialize: create a new Git repository
2. Add: add a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit.
3. Commit: save your changes to the local repository
4. Pull: fetch and download content from a remote repository and immediately update the local repository to match that content
5. Push: upload local repository content to a remote repository.

Some advanced Git operations are[3]:

1. Branching: a Git branch is an independent line of development. You can take advantage of branching when working on new features or bug fixes because it isolates your work from that of other team members.
2. Merging: integrate changes from another branch. Merging takes the contents of a source branch and integrates them with a target branch. In this process, only the target branch is changed. The source branch remains the same.
3. Rebasing: compresses all the changes into a single "patch." Then it integrates the patch onto the target branch. Unlike merging, rebasing flattens the history because it transfers the completed work from one branch to another. In the process, unwanted history is eliminated. Rebases are how changes should pass from the top of the hierarchy downwards, and merges are how they flow back upwards

So why should we start using Git? Let's start with the advantages[4]:

- Has another researcher/author already written code that you could use on your project? If so, Git allows you to access that code, adapt it for your own project and collaborate with the original author.
- Distributed model: your work is your own. You can let others see only what is necessary. Not everything has to be public.
- Branching and merging are easy: branching and merging feel like a natural part of the workflow. Branching is very fast and consumes very little space, so you can branch whenever you want and isolate your new features and ideas until they are ready for the mainstream.
- Workflow is flexible: Git allows you to choose your own workflow. It can be as simple as a centralised workflow or as hierarchical as the dictator-lieutenant workflow. Use the process that best fits you.
- Data integrity is assured: Git tracks all the files and directories of your project in such a way that it is not possible to introduce unnoticed changes.
- Other advantages are:
    - Fast: Git is very fast, even when compared to other DVCS, for local as well as network operations
    - Staging area: staging is a step before the commit process in Git. In this way you can make sure that your commits have logically grouped changes.
    - Free: Git is free and open-source. There are license fees.

Disadvantages[4]

- Steep learning curve: many commands with many options, some commands are non-intuitive and need a level of understanding.
- Binary files are a big no: If your project has non-text files that are updated frequently (images or MS Office documents) Git becomes bloated and slow.

Companies and Technologies around the world use Git: Amazon, Facebook, LinkedIn, Yahoo, Microsoft, Netflix, Rails, Android, Linux and Zendesk — just to name a few. Just learn Git. You won't regret it. Although it has many options you will have the basics down in 15 minutes, and within a couple hours you can be making pull requests to open-source projects. [6]

Mark Bynens (J&J)

Markus Elze (Roche)

Lyn Taylor (Phastar)

Written on behalf of PSI AIMS

[1] Chou, Eric. "Mastering Python Networking: Your one-stop solution to using Python for network automation, DevOps, and Test-Driven Development, 2nd Edition", Packt Publishing Ltd., August 2018, P.345.
[2] Resham Akhtar Ahmed, Always "Git"ing ahead, May 24, 2017, https://www.quora.com/What-is-Git-and-why-should-I-use-it

[3] Vali Shah, Nov 14, 2018, https://medium.freecodecamp.org/an-introduction-to-git-merge-and-rebase-what-they-are-and-how-to-use-them-131b863785f

[4] Sahyadri Holagundi, Want to love git, but it is not letting me, Nov 22, 2015, https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-Git

[5] Nic Ryan, Why Git?, September 25, 2018, https://www.datafriends.rocks/single-post/Why-Git

[6] Brandon Morelli, New Developer? You should've learned Git yesterday**,** Jul 5, 2017 https://codeburst.io/number-one-piece-of-advice-for-new-developers-ddd08abc8bfa