

# Getting started with version control for reproducible science

Brad Duthie

13 January 2023

## Contents

---

After reading through this, you should have a working understanding of what version control is and why it is useful. You should also be able to start using version control in your own work through GitHub and GitHub Desktop .

---

- Introduction: What is version control?
  - Getting started with GitHub
  - Installing GitHub Desktop and Tutorial
  - GitHub Interface & Command Line Interface (CLI)
  - Initialising and using a git repository
  - Some additional points
  - Additional resources
- 

## Introduction: What is version control?

Version control is any system that records changes made within a set of files over time so that different versions of files can be managed and, if necessary, recovered. **Put more intuitively**, version control is a way of taking a snapshot in time (called a ‘commit’) of all the files in one of your folders (called ‘repositories’); as you make changes to the files within your folder, you can always come back to previous snapshots that you’ve taken (if, e.g., you make a change that you regret, or need information from a previous point in time). You can even have multiple different versions of the same folder existing in parallel (called branches). You can think of it as an extra step on top of ‘saving’ a file – a step that solidifies a key point in time for your work, records how it changed from previous and subsequent points in time, and records who made the change, when, and why.

Version control is indispensable for large coding projects with multiple developers collaborating on the same code, but it’s also a very useful tool for the workflow of scientific research. Using version control can allow you to better manage data files, analysis files (e.g., R code), manuscript files, and more in a way that keeps things clean and removes the anxiety of losing track of which file is the ‘right’ one.

Version control is also an excellent tool for doing open science. By keeping a record of how your data, analysis, and manuscripts change over time, the process of doing science becomes more transparent. By





















<b>Add notebook files and vinettes</b> bradduthle committed on 20 Aug	 a182224	
<b>Add lines and edits to SI -- finish all edits for Reviewer 1 and add ...</b> ... bradduthle committed on 20 Aug ..documentation	 00313cf	
<b>Rework SIs and MS with new plotting function</b> bradduthle committed on 20 Aug	 34f87e9	
<b>Update versoin</b> bradduthle committed on 20 Aug	 39085dd	
<b>Change plotting so that the key plots only take one argument</b> bradduthle committed on 20 Aug	 0348071	
<b>Update GMSE version number</b> bradduthle committed on 20 Aug	 cf3b881	
<b>Edit based on roxygenise</b> bradduthle committed on 20 Aug	 d95c8b0	
<b>Address Reviewer 1 minor comment 5 with a new section of SI3 on non-d...</b> ... bradduthle committed on 20 Aug ..efault parameter values	 1d7fb25	
<b>Fix documentation for mark-recapture analysis</b> bradduthle committed on 20 Aug	 c31493c	
<b>Update manual files</b> bradduthle committed on 20 Aug	 be413ed	

Figure 1: \*An example of the timeline of commits for a recent project. Bold titles on top show more recent changes committed to the repository; the bold messages are written at the time of committing, and make it easier to see important changes added over time. In GitHub, you can click on these bold titles to see what changes were made since the last commit; from here, you can also see the whole repository as it was during the time of commit (you can also do this by clicking on the '<' '>' buttons on the right).\*

uploading your progress to GitHub, you can make the whole process of doing science accessible to others, and have evidence of priority and accuracy in your conclusions (you can also keep repositories private).

There are many different types of version control available. Here, I am going to focus only on git version control software, which has the advantage of being free, open source, available on all platforms (Linux, Mac, and Windows), and the most popular software among research scientists. The software was invented by Linus Torvalds, the same developer who created the Linux kernel.

In this introduction to using version control, I am going to focus heavily on using two software tools that work with git, GitHub and GitHub Desktop. Like git, both GitHub and GitHub Desktop are free for basic use, though more advanced options can come with a small cost. These two tools make using git much easier, especially if you don't like the idea of working within the command line. GitHub offers a massive online platform where you can store your git repositories, discover and download new repositories, and collaborate with other GitHub users (e.g., in organisations such as the Stirling Coding Club). GitHub Desktop provides a nice graphical user interface for using git, visualising your repository, and linking to GitHub. As you become proficient with git, you might find yourself start thinking less in terms of individual files and file versions, and more in terms of commits and branches with inter-related files.

First, I am going to briefly talk about how to use git entirely *within* GitHub on your browser. This requires fewer steps than using GitHub Desktop, but it can only get you so far because you cannot work with the changes that you make directly on GitHub. For example, if you edit an R file in GitHub, you would have no way to run the code without pulling the file from GitHub to your local repository.

## Getting started with GitHub

If you are just starting out with git, I recommend signing up with a GitHub account, then downloading GitHub Desktop. You can do everything that I explain below with only these two tools. For those who prefer to use the command line interface, I have included instructions for how to do this below too. Learning to use git in the command line is probably useful if you already use the command line in your normal work flow (or if you are interested in doing so!), but if it's not something that you work with already, learning it here is probably more trouble than it's worth.

## Downloading the GitHub Desktop

Once you have a GitHub account, you can download GitHub Desktop. This is not technically necessary to do to use git and GitHub. The desktop application only works with Windows and Mac (which is part of the reason I use the command line and am including the CLI instructions here). You can find the link (<https://desktop.github.com/>) by scrolling down your home screen on GitHub. The link will take you to the page below.

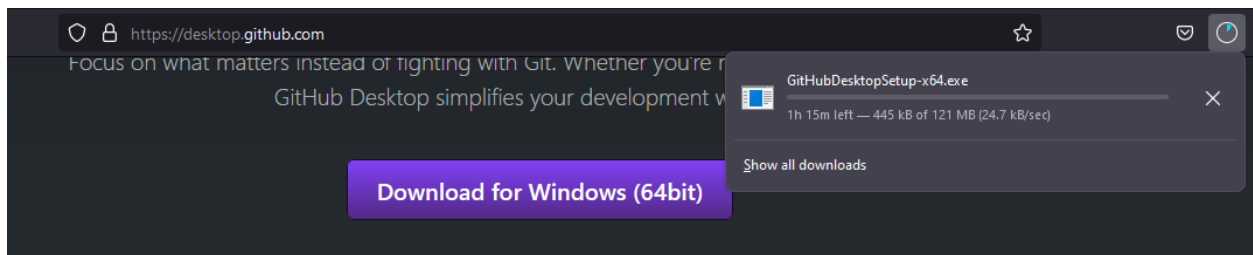


Figure 2: \*The GitHub Desktop download website.\*

After GitHub Desktop has downloaded, you should be able to find the downloaded file in your downloads folder.

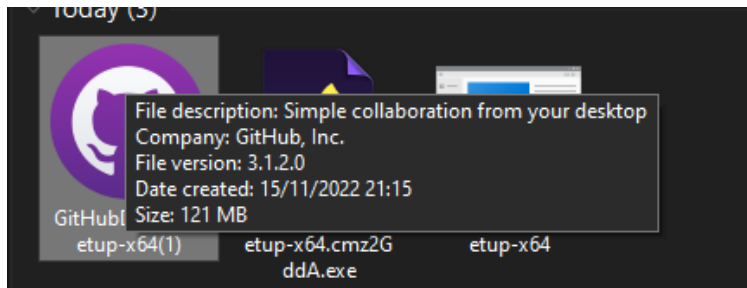


Figure 3: \*The downloaded executable file should show up with an icon in your download folder.\*

Double click on the downloaded file, then GitHub should install.

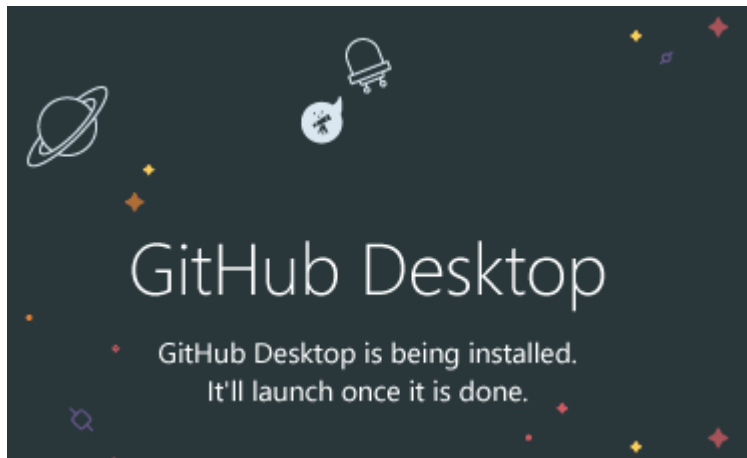


Figure 4: \*GitHub Desktop is being installed.\*

Once GitHub is installed, you should sign into your GitHub account. It is important to do this to link your GitHub account with the GitHub Desktop.

You will be asked to authorise the desktop application to access GitHub. Again, this is important to do so that you can link your GitHub account with GitHub desktop. Once you do this, you can configure your GitHub account. Here is mine below.

That should be it! You should now have your GitHub Desktop linked with your GitHub account online. The tricky part is getting started and getting into a rhythm of using git and GitHub. The tutorial is a good place to start, although I think that it complicates things a bit unnecessarily by introducing the concept of branching very early on. Nevertheless, I still recommend first going through the (brief) tutorial as a starting point.

The tutorial will create a new repository (i.e., a folder) on your computer (probably in 'Documents/GitHub'). This repository has another (hidden) folder within it that holds all of the information on the history of the repository. It is also initialised with a file called 'README.md', which you can open with any text editor (e.g., Rstudio). The interface of the GitHub Desktop is shown below.

The first task that the tutorial will have you complete is making a new branch. Go ahead and do this, but it is actually not strictly necessary. There is nothing wrong with just working off of the main branch, and we will get to what branches are later (for getting started, and for a lot of lone work, they can actually kind of be ignored). The rhythm that you really need to get comfortable with is editing files, committing them, then passing them to GitHub. The next task that the tutorial has you do is edit a file by adding a new line to the file 'README.md'. You can do this in Rstudio, or any text editing software (e.g., notepad or gedit, but not a Word Processor such as MS Word or LibreOffice).

# Welcome to GitHub Desktop

GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise. Sign in below to get started with your existing projects.

New to GitHub? [Create your free account.](#)

Sign in to GitHub.com

Sign in to GitHub Enterprise



Figure 5: \*Welcome to GitHub Desktop.\*

## Configure Git

This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.

- ☒ Use my GitHub account name and email address  
☐ Configure manually

Name

Brad Duthie

Email

brad.duthie@gmail.com

Finish

Cancel

Example commit

Fix all the things

Brad Duthie • 30 minutes ago

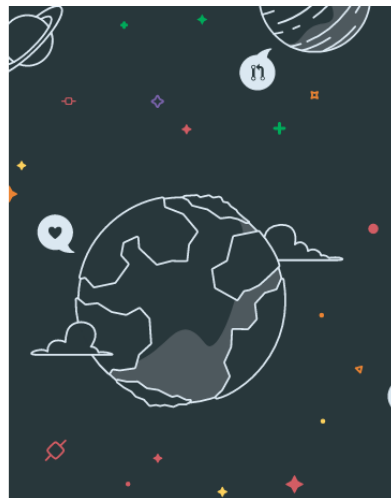


Figure 6: \*Make sure to configure your GitHub Desktop.\*



Create a tutorial repository...

Figure 7: \*Start out with a tutorial repository.\*

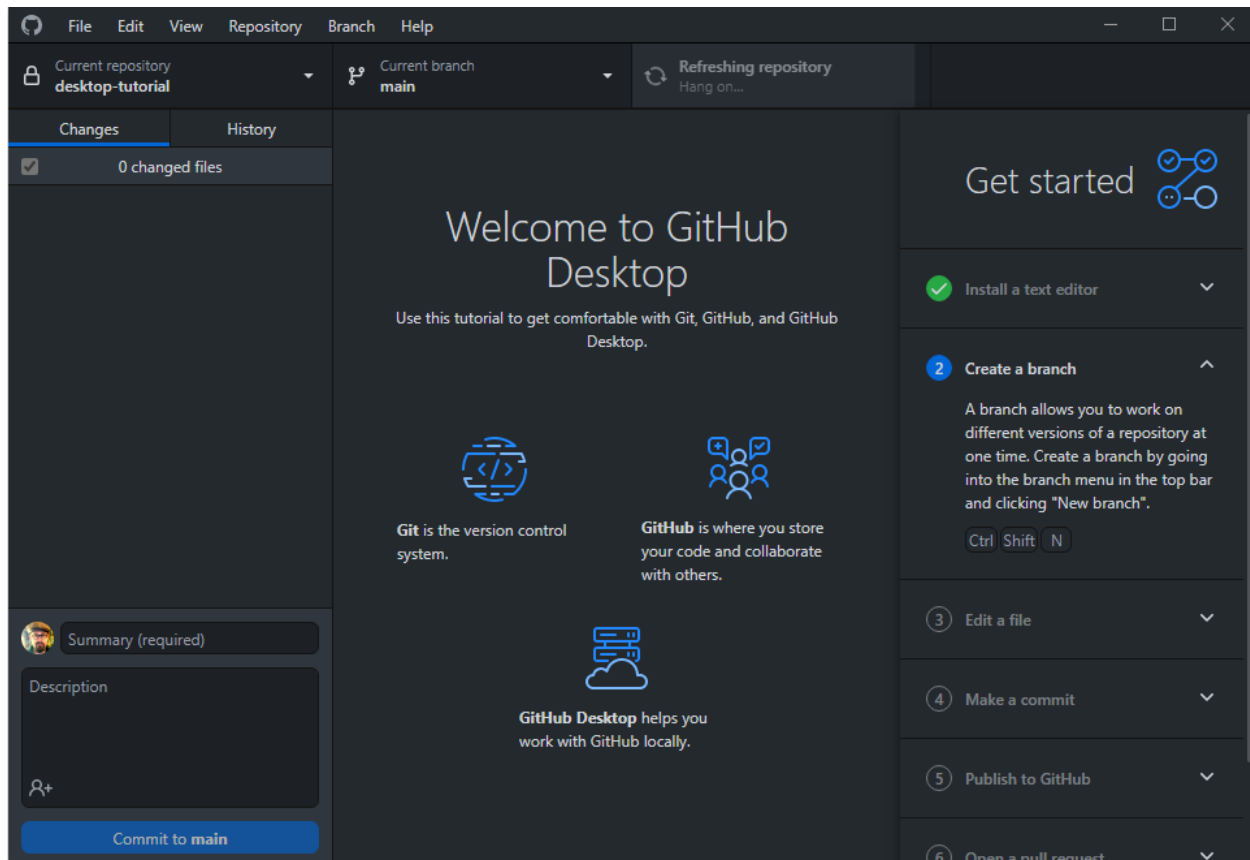


Figure 8: \*Welcome to the tutorial for GitHub Desktop\*

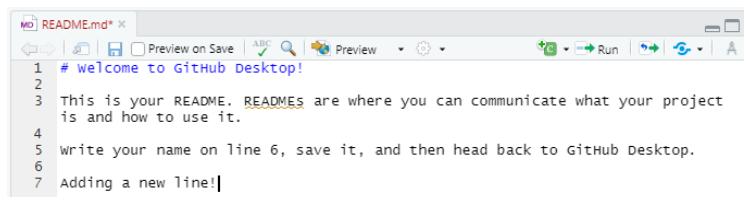


Figure 9: \*Add a new line to the README.md file in the GitHub Desktop tutorial\*

Save the README.md file after changing it somehow. When you go back to GitHub Desktop, you will then see the file again, but with the lines that you just added highlighted in green (lines removed would show up in red).

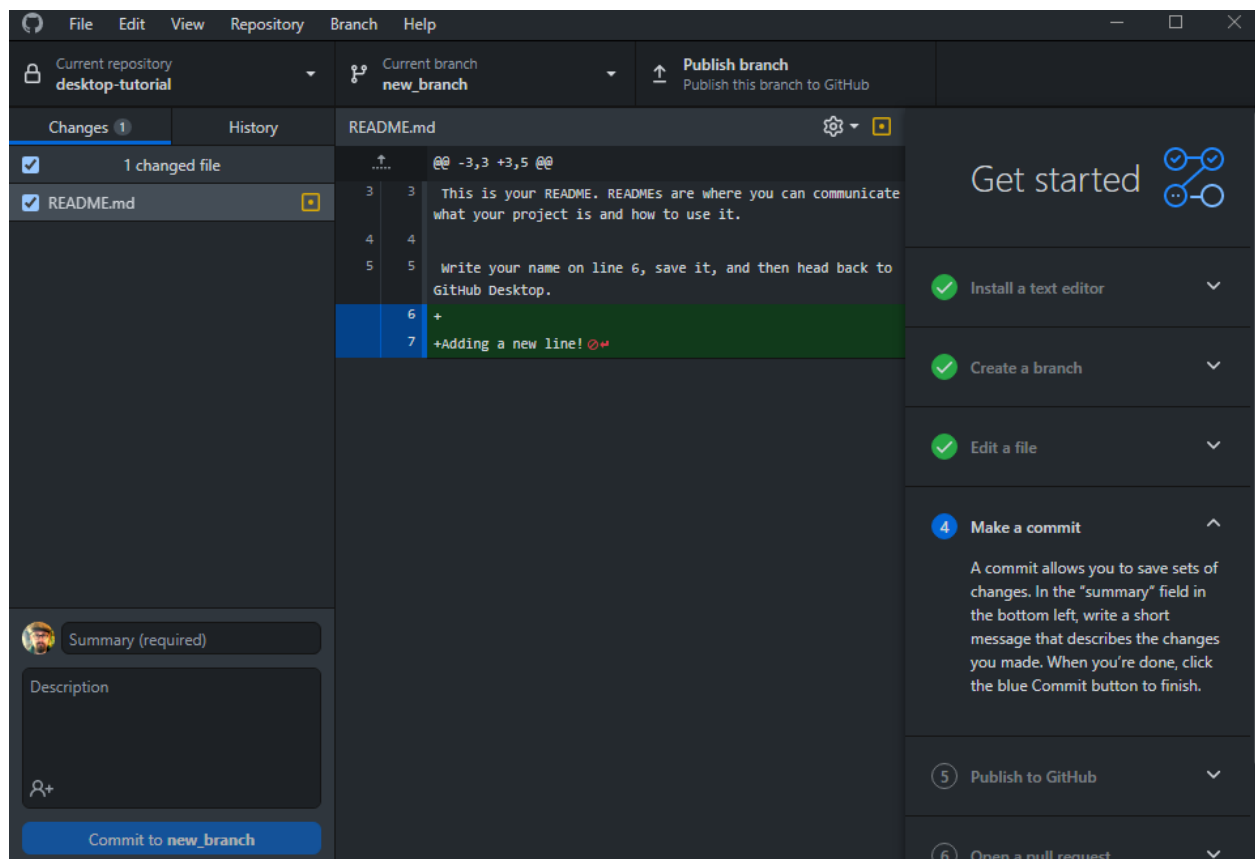


Figure 10: \*A new change is shown in the GitHub Desktop, with the option for making a new commit.\*

What git is doing here is comparing the current state of the file to the most recent committed version of it (i.e., the original initialised version, in this case). This can be very useful for reminding yourself what you have saved since your last commit, and you can see this comparison on the GitHub website too for the entire history of your repository. In other words, at every commit, it is possible to view what has changed from the previous commit. Note that this is only true for plain text files (e.g., files with extensions of R, Rmd, TXT, CSV, MD), not binary files that cannot be viewed as plain text (e.g., DOCX, XLSX, or any image files).

Note in the lower left of the figure above that we have the option to create a new commit. This option is always available when some file that is being tracked in the repository has changed. It is good practice to commit changes regularly, which makes it easier to see the history of a repository and backtrack to any place in its development. It is also good practice to commit with an informative message, so that when you or someone else scrolls through the commit history, it is clear what changes have been made with each new commit.

Once we have committed the change to our new branch, we need to *push* it to GitHub (we can also *pull* from GitHub, but we will worry about this later). The tutorial will show you how to do this, then show you how to make a pull request and merge the changes from your new branch into the main branch. This will be easy to do because we have not changed the main branch at all, so there will not be any conflicting changes when merging. When there are conflicting changes (e.g., suppose you made changes to the main branch, then made *different* changes to your new branch and tried to merge the two), you will get a merge conflict. This is not a cause for concern; it just means that you need to figure out which changes should apply to the merge.

## GitHub Desktop

Now that you have completed the tutorial, here is how I recommend that you start using git and GitHub in your day-to-day work. If you have heard about branches and merges, you can just forget about them for now. Just focus your workflow on saving, committing, and pushing. This will create a clear history of your project repository that is saved on your computer and on GitHub, and you can always rewind back to a previous version.

Throughout this section, I will explain how to use the basic functions of git and link git with GitHub using GitHub Desktop. Note that GitHub Desktop can run alongside whatever text editor (e.g., Rstudio) you're using to actually edit your text files, but you do not typically make edits from within GitHub Desktop itself – GitHub Desktop is more like a file manager in this way; you can keep it open to commit and push while working on files in your git repository.

## Initialising and using a git repository

Initialise a git repository in the GitHub Desktop by navigating to File and choosing 'New repository...' from the pulldown menu. You can do this every time you want to start a new project file, and I recommend that you keep all of the files relevant to a specific project (e.g., a thesis chapter) in the same repository (i.e., data, R scripts, writing).

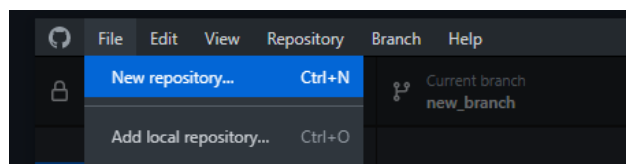


Figure 11: \*Create a new repository in the GitHub desktop.\*

Make sure to name the project something descriptive and choose a path on your computer where the repository will be held. Where you choose to put your repositories will depend on your preferences for file organisation, but I prefer to have a folder called 'projects' on my computer that has all of my repositories for scientific projects. Again, I recommend organising your computer folders by project, not saving different project components such as data, analyses, and writing all in different places (this will get confusing when you have multiple projects, I promise!).

Once you have created your new repository, immediately publish the repository to GitHub to link between your computer and GitHub.

Now you are ready to go. Every time you make some substantial changes to files in your repository, use the GitHub Desktop to commit the changes to the main branch, then push them to your repository on GitHub. This will seem like an extra unnecessary step as you work, at first, but I promise you that it will, at some point, save you a lot of time and frustration. The common wisdom in using version control is “**commit early, commit often**”, meaning that it is generally better to err on the side of committing changes that you make to your repository more often than you think is necessary. In practice, I try to commit frequently, almost as often as I save file changes; even small changes can be useful to have their own commit because the changes are easier to read in small chunks rather than massive changes to files.

You will also find that your mindset for working will change a bit; the entire history of your project will be recoverable, so you can be a bit more bold with trying new things and making changes. To get a sense for what all of this looks like, please feel free to look through the commit history of some of my projects (and feel free to ask questions).



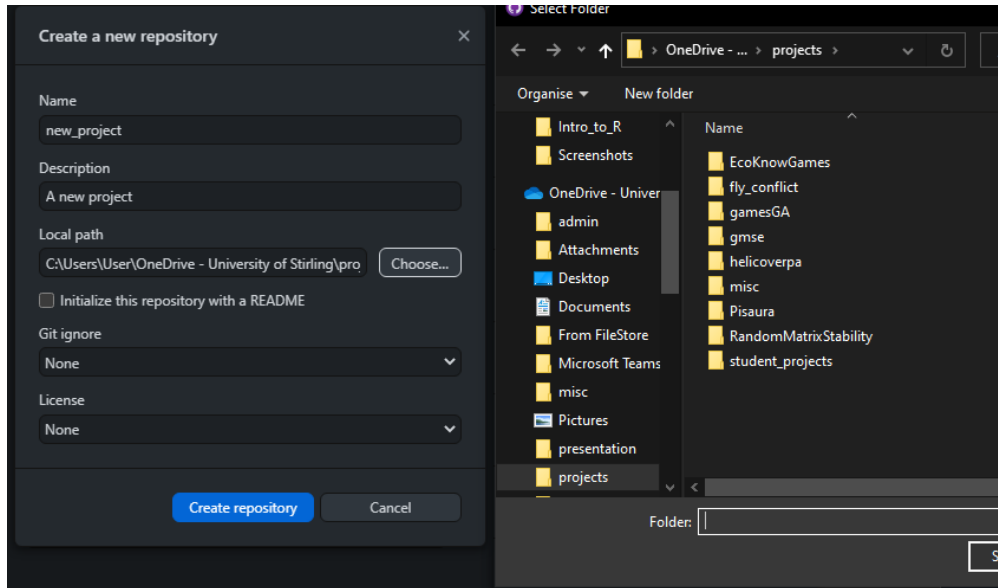


Figure 12: \*Create a new repository in the GitHub desktop with an informative name and memorable location ('Local path').\*

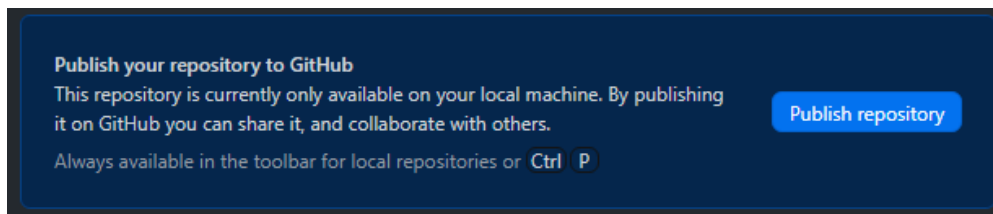


Figure 13: \*Publish your repository to GitHub as soon as you can.\*

## Some additional points

Remember that you can see the changes made to plain text files (e.g., files with extensions of R, Rmd, TXT, CSV, MD), not binary files (e.g., DOCX, XLSX, or any image files). This does not mean that you cannot commit and push changes to binary files using git; you absolutely can and should!

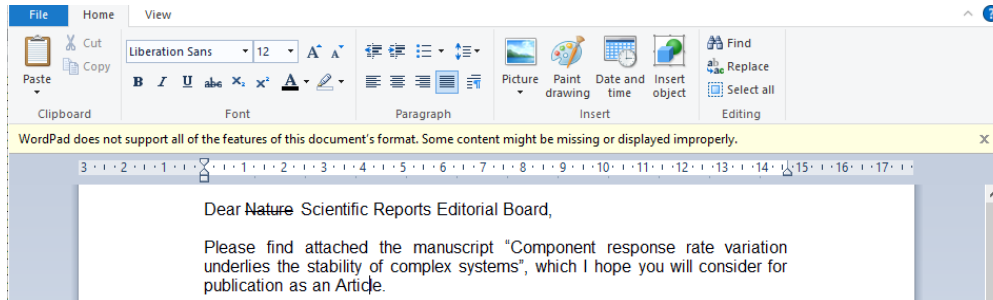


Figure 14: \*Binary files still work with git, but you cannot see line by line changes.\*

But if you make a change to a binary file, such as the cover letter written in a word processor above, then git will not be able to show you line by line changes. You will just get a short message in the GitHub desktop.

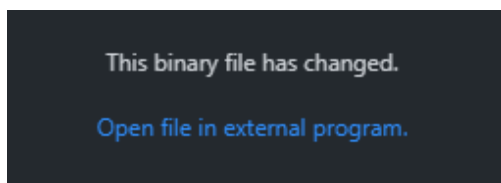


Figure 15: \*Binary files changes are not shown in GitHub Desktop.\*

There is a lot more to consider. These notes are an abridged and updated version of a longer set of notes on version control that I wrote for a previous IAPETUS workshop and Stirling Coding Club session. The longer version explains some additional features of git, including pulling, branching, and merging (including dealing with merge conflicts). Please feel free to check these out. There will be another special session on version control as part of Stirling Coding Club on 11 January 2023 over MS Teams (all are welcome). The longer notes also use the command line interface instead of GitHub Desktop, and also a program called GitKraken, which used to be free, but now costs 5.00 GBP per month after a 7-day trial.

As you get more comfortable with git, you can start exploring new features such as branching, which allows you to create new offsets of your main branch and make changes without disturbing it. This is extremely useful if you have some code that is working, or a draft of a manuscript that you really like, but you want to try something new without being afraid of breaking what you already have. Using GitHub, you can also code collaboratively, so you and your colleague might work on different parts of a repository on different branches, then merge your work later on down the line. In my own projects, I usually have somewhere between 2-5 branches at a time, including one main branch that I know is stable, a dev branch that is ready for publication with just a bit more review, a branch that I actually make changes on, and multiple experimental branches in which I am trying new things.

I strongly encourage you to try this simple workflow of saving, committing, and pushing for a while. You will not break anything! If you have questions, get stuck, or start getting nervous, you are welcome to get in touch with me by email (alexander.duthie@stir.ac.uk), by MS Teams, or on GitHub (<https://github.com/bradduthie>). Note that you can invite people into your repositories on GitHub and tag them as you would with other social media (e.g., @bradduthie). You can even comment directly on lines of code in GitHub.

## Additional resources

### Introductions to version control (guides)

- Introduction to Version Control for Stirling Coding Club
- British Ecological Society: A guide to reproducible code in ecology and evolution
- An introduction to version control

### Introductions to version control (videos)

- Git and GitHub for Poets: Intro (no programming knowledge needed)
- Git and GitHub for Poets: Branches
- Git and GitHub for Poets: Fork and Pull