Introduction to Git and Github

1 Introduction

In this short course, we'll be using the version control program 'git'. You'll learn what version control is, how to build and update a repository and how to use GitHub.

1.1 What is version control?

If you were to look at some of my old projects on my computer, you'd see a few things:

- Everything is inside a single folder, with little structure
- There are filenames like "ArticleDraft7_B_EDIT_THIS_ONE_(Backup).doc"
- There are filenames like "DataFromBob_fromEmail EDITED.xls" which had been emailed back and forth four times
- The draft with the highest number is not the one that was edited last
- There are R scripts that point to files that no longer exist on my system
- Some of the functionality relies on me remembering not to run particular lines in a script

In other words, its *disorganised*. If I or someone else wanted to pick up the project again, it would be very difficult to figure out where to start. It's also very redundant - I'm storing whole copies of files with just a few differences between them.

The answer to this mess is version control. The idea is that you store *changes* to files in a way that you can go back to the way a file was at a preivous time. It also makes it easy for different people to work on different bits of the project at the same time. By comparing the differences between people's versions, it's possible to merge changes in an organised way.

Using version control means that you can have a single file for the most up-to-date version of your article manuscript, but still keep the entire history of edits. It also basically forces you to make sure your project is well organised and documented.

There are lots of examples of programs that do version control. In this tutorial, we'll be using the program **git**, but programs like Mac's Time Machine or Google Docs or WriteLatex/Overleaf also use version control.

1.1.1 Basic concepts of version control

Git stores a **repository** of information about your project. You select files for the repository to **track**, then edit the file as usual. When you've made a significant change, you can **commit** the changes. At this point, all the differences between the last commit and the current file will be stored in the repository. That means you can **checkout** different versions of the file at different points in its history.

The general cycle of using version control is the following:

- Initialise a repository to track changes
- Add files to track
- Commit changes to the repository
- Make changes
- Commit changes to the repository …
- Make changes …
- Push repository to an external host

Pretty much all the code you'll learn in this tutorial is here:

```
> git init
> git add *
> git commit "first commit"
> git add *
> git commit "changed title"
> git push
```

1.1.2 WARNING

Version control is not the same as backing up data! Git will store the history of your project, but if your computer is damaged or lost then you also lose your project. Backing up to another hard drive / cloud storage is also necessary

1.2 Basic setup for this course

Before the course, you need to do 3 things: install git, install a modern text editor and get a free github account. The basic intro in this document will show you how to do this, and also show how to navigate directories at the command line. Finally, we'll set up some files that we'll use in the rest of the tutorial.

1.2.1 Installing git

You can download git here: https://git-scm.com/downloads

If you're on Windows, just run the installation program and select all the default settings in the install wizard. You should end up with a program called GitBash on your machine.

If you are familiar with the command line or have a mac, the link above will work fine, but there are also some more details about other ways to install git here). (I use git through the Mac Terminal, and installed it from the command line).

For the moment, I would recommend avoiding GitHub desktop.

1.2.2 Installing a modern text editor

You should also install a modern text editor, such as:

Windows: Notepad ++

Mac: [Text Wrangler] (http://www.barebones.com/products/textwrangler/download.html) or [Sublime] (https://www.sublimetext.com/)

Most Linux distributions come with a decent text editor

Note that Notepad for Windows or TextEdit for Mac can corrupt text files.

1.2.3 Getting a github account

You should sign up for a github account. It's free. Go to: https://github.com/ and click 'Sign up'.

1.3 Make some folders and files

Make a folder for this Introduction. Inside that folder, make a folder called `tutorial1`.

Make a new text file called `animals.txt` with the following text:

```
Donkey
Dolphin
Monkey
Baracuda

```

Note the blank last line

Save `animals.txt` to the `tutorial1` folder.

1.4 Navigating directories in the terminal (*cd*, *pwd* and *ls*)

This section introduces basic navigation commands such as `cd`, `pwd` and `ls`. If you're familiar with these, you can skip to the next section.

Start up your terminal (Mac/Linux) or GitBash (Windows). These use standard Linux bash commands.

This is where you can type commands to run programs like git. Be careful - the terminal is very powerful and some commands can delete files. But if you follow this tutorial, you shouldn't run into any problems.

At any point, the commands you type will apply to the **working directory** ("directory" means the same thing as "folder"). To find out what the current directory is, type `pwd` (print working directory). The box below shows the command you should type. The `>` symbol represents the command prompt, and you don't have to type this character. Press enter after each line. Any line that does not have a command prompt in front of it is output - you shouldn't type this.

```
> pwd
```

Type this and press enter. For me, the pwd command returns `/Users/sgroberts`. For you, it will probably be your home directory. We want to set the current directory to the `tutorial1` folder. You can **change directory** using the `cd` command, followed by the location of the directory you want to change to. On my computer, the `tutorial1` folder is here:

```
> cd ~/Documents/Teaching/IntroToGitHub/TutorialFolders/tutorial1/
```

The `~` character at the start is a shortcut for "my home directory".

Now we can check the working directory again:

```
> pwd
/Users/sgroberts/Documents/Teaching/IntroToGitHub/TutorialFolders/tutorial1
```

If we're navigating just one folder up or down, then there's no need to type the whole thing. To move one directory up, type:

```
> cd ..
```

And to move into a sub-directory of the working directory, just use the name of the directory

```
> cd tutorial1
```

You can get a **list of files and folders** inside a directory by using `ls`:

```
> ls
animals.txt
```

The `tutorial1` folder has 1 file inside called `animals.txt`.

1.4.1 Go on to the next section