Using Git and GitHub for collaborative research

What, how and (most importantly) why?

Jon Minton

[Jonathan.minton@glasgow.ac.uk](mailto:Jonathan.minton@glasgow.ac.uk)

10 July 2014

# Introduction

This document will describe Git/GitHub, and discuss how it can help AQMEN researchers with a range of important and related tasks including data management, code management, collaborative working, archiving, and producing fully reproducible research.

# What is Git?

Git is described as ‘a free and open source distributed version control system designed to handle everything from small to very large produce with speed and efficiency.’[[1]](#footnote-1) Git is a command-line based software tool, developed by Linus Torvalds, the creator of Linux, in order to make the daily management of complex, large scale software projects both simpler and more robust. Although it has its origins in computer science, the potential applications and user base is much broader, and has great potential within quantitative social sciences.

In practice, Git can be thought of a file management system that takes repeated ‘snapshots’ of files at various stages in development, and stores these snapshots in online repositories. Other ‘cloud’ management tools like Google Drive and Dropbox also provide this capability to some extent, in that earlier versions of most files are stored and it is possible to ‘roll back’ to earlier versions of files in many cases. However, Google Drive and Dropbox typically only store earlier versions of files for a limited period, around 30 days. By contrast, earlier versions of Git files can be stored indefinitely, providing a complete record of the development of code, data and other written documents. Git also enforces ‘literate’ file archiving: the user must specify when Git should take document snapshots, and must include comments very briefly describing how and why they have modified the documents. Because of this, the project development audit trail is not only complete but also comprehensible.

Git provides a number of highly advanced options for managing complex projects involving many documents, file types and users. For example, it allows multiple users to work on their own version of a particular code or syntax file at the same time, and reports how the versions differ from each other, and what ‘conflicts’ exist within the file that have to be ‘resolved’ in order to be coherently merged back into a final, agreed-upon version. This kind of fine grained version control is only designed to work with text files, which of course includes almost all STATA, SPSS or R syntax files, and allows people to work in parallel on a project without parallel, subtly different and incompatible, versions of the projects developing.

A related advanced feature of Git is that it allows users to create parallel versions of a project on purpose, rather than by accident, by creating ‘branches’ of a single project which can be archived, recoded and developed in parallel, before being merged back or ‘chopped back’ later. This may be useful, for example, if you want to see if a different approach to managing some data and running a statistical model may be more effective than the current approach, but doing this would involve deleting large sections of the code and starting again. Using this approach, you could choose to create an Alternative branch, in which you try to develop the alternative approach, as well as the Original branch, where you keep developing using the existing approach. It is relatively straightforward to switch between branches, making many incremental developments to both branches, each with distinct and independent archiving and version histories, before deciding which branch, or which pieces of either branch, you want to retain within the project. These branching and merging capabilities allow researchers to experiment and try out new approaches without losing gains that have already been made.

# What is GitHub?

Git requires online repositories, in which all snapshots and versions of the project are stored. GitHub is a very popular online Git repository, but also provides some additional features of its own. Firstly, it has an elegant and easy to use online interface for accessing files, showing changes and information about changes (‘commit messages’), viewing file and folder revision histories, and so on. Secondly, it includes a GUI program, which can be downloaded onto PC or Mac, which allows users to use Git without having to learn the Git command line interface. These two additions make the underlying features of Git much more accessible and user friendly, opening up its potential to a much wider audience.

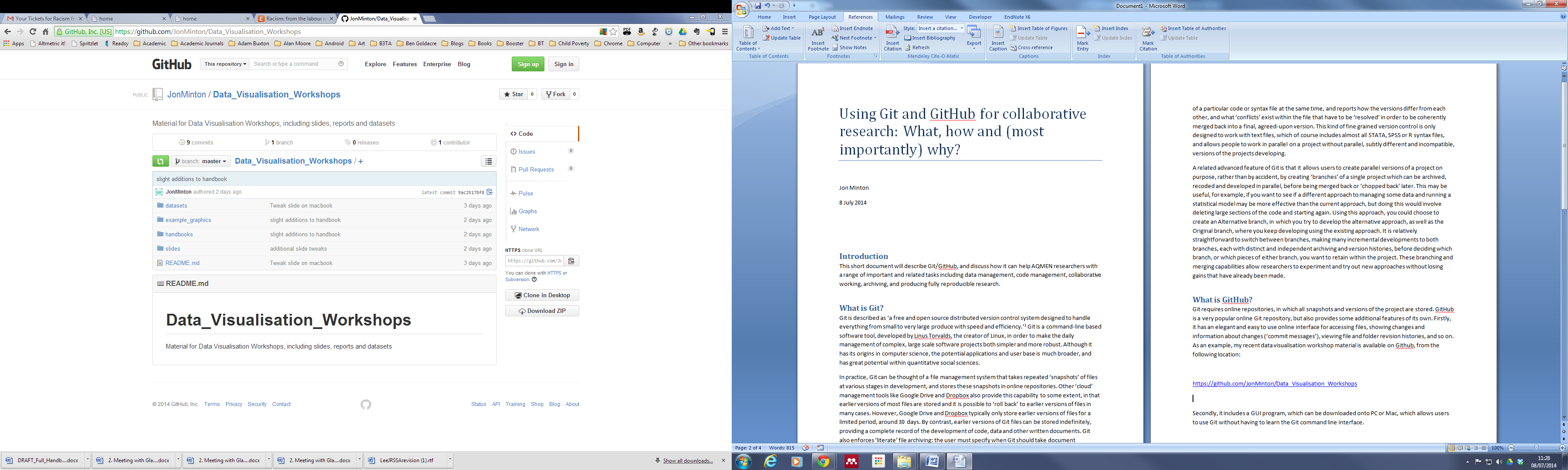
# An example: using GitHub for general file management

GitHub and Git are intended to be used for text files, i.e. files that can be read by humans if opened in something like Notepad. Powerpoint presentations, word documents, and data stored in an SPSS, Stata or R native format are known as ‘binary’ files, as they need a particular type of software to interpret properly. Git and Github were not designed around such files, and the advanced branching, merging, and simultaneous collaboration features described earlier will not work as well when using them. Nevertheless, GitHub turns out to be a highly effective system for archiving, developing and maintaining binary files as well as text files.

As an example, my recent data visualisation workshop material is available on Github, from the following location:

<https://github.com/JonMinton/Data_Visualisation_Workshops>

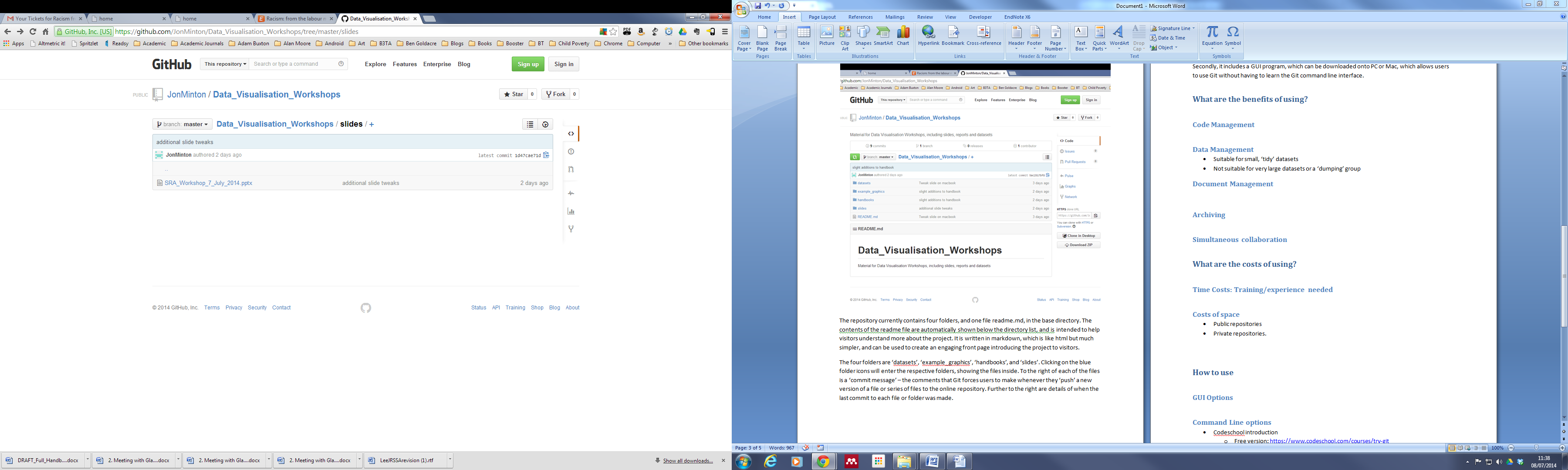
A screenshot of the website is show below.



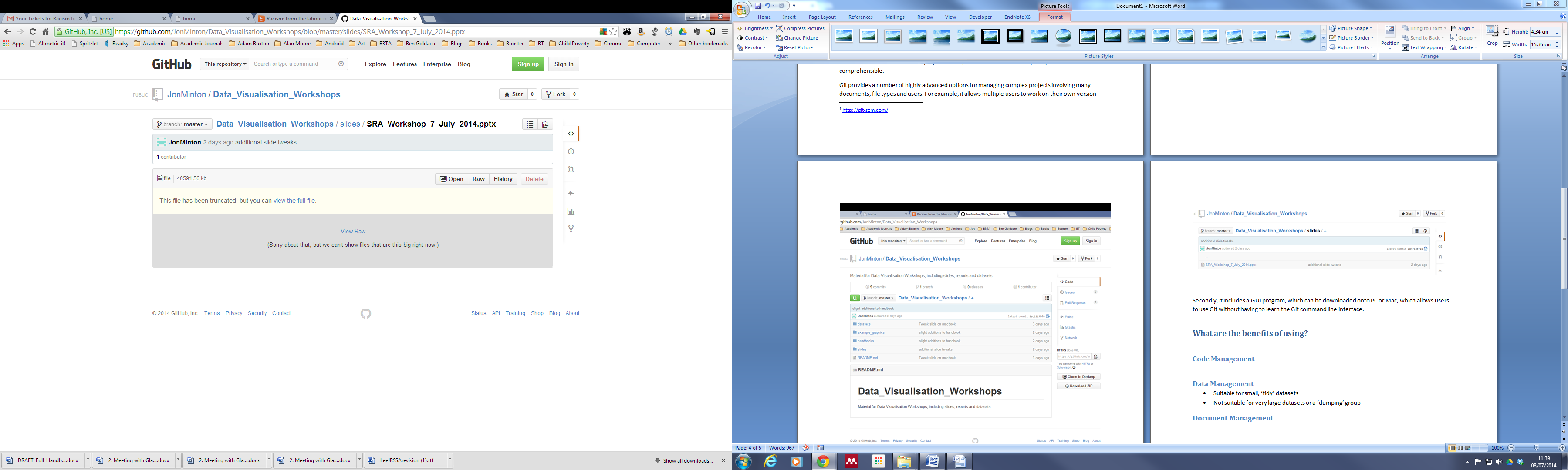
The repository currently contains four folders, and one file readme.md, in the base directory. The contents of the readme file are automatically shown below the directory list, and is intended to help visitors understand more about the project. It is written in markdown, which is like html but much simpler, and can be used to create an engaging front page introducing the project to visitors.

The four folders are ‘datasets’, ‘example\_graphics’, ‘handbooks’, and ‘slides’. Clicking on the blue folder icons will enter the respective folders, showing the files inside. To the right of each of the files is a ‘commit message’ – the comments that Git forces users to make whenever they ‘push’ a new version of a file or series of files to the online repository. Further to the right are details of when the last commit to each file or folder was made.

Clicking on the ‘slides’ folder shows the contents of the folder:



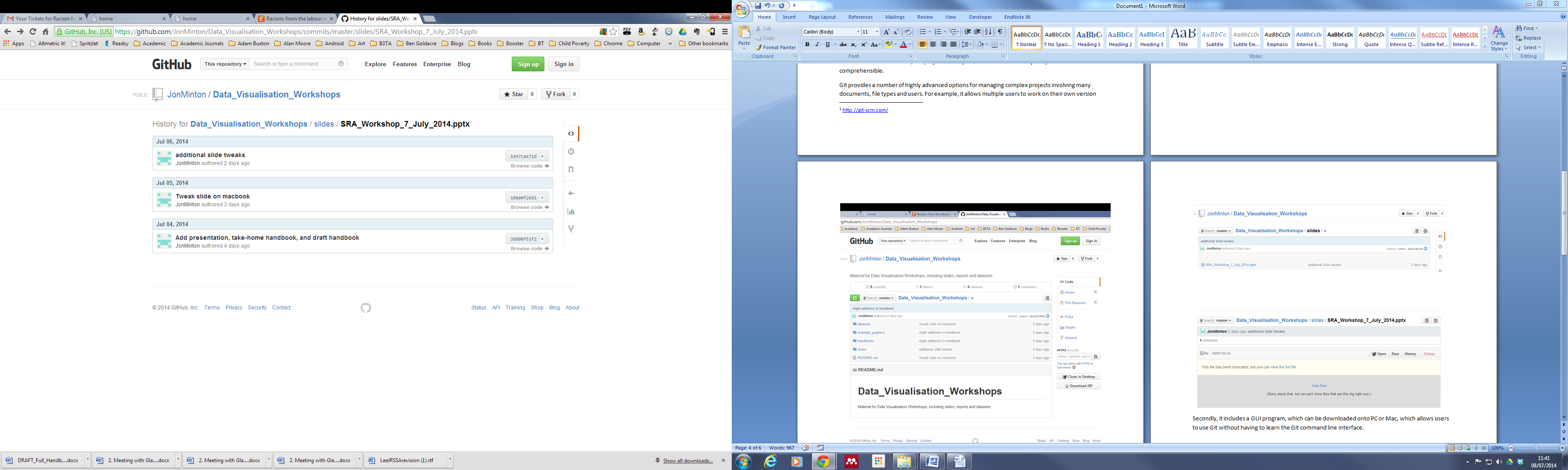
In order to access the file, SRA\_Workshop\_7\_July\_2014.pptx, first has to be clicked on, producing the following:



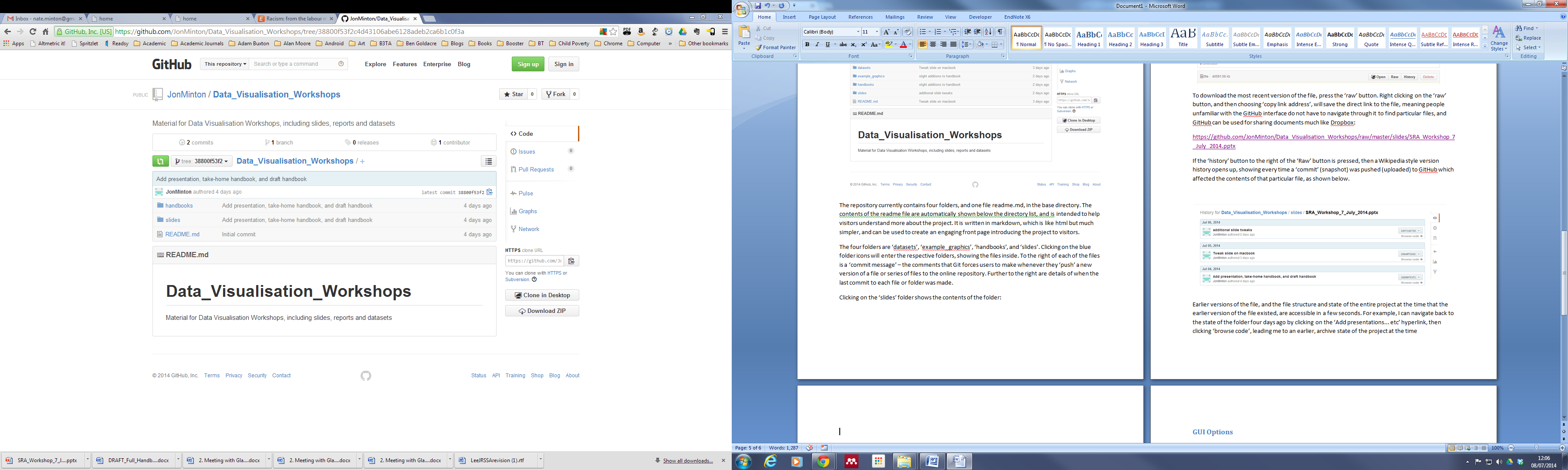
To download the most recent version of the file, press the ‘raw’ button. Right clicking on the ‘raw’ button, and then choosing ‘copy link address’, will save the direct link to the file, meaning people unfamiliar with the GitHub interface do not have to navigate through it to find particular files, and GitHub can be used for sharing documents much like Dropbox:

<https://github.com/JonMinton/Data_Visualisation_Workshops/raw/master/slides/SRA_Workshop_7_July_2014.pptx>

If the ‘history’ button to the right of the ‘Raw’ button is pressed, then a Wikipedia style version history opens up, showing every time a ‘commit’ (snapshot) was pushed (uploaded) to GitHub which affected the contents of that particular file, as shown below.



Earlier versions of the file, and the file structure and state of the entire project at the time that the earlier version of the file existed, are accessible in a few seconds. For example, I can navigate back to the state of the folder four days ago by clicking on the ‘Add presentations... etc’ hyperlink, then clicking ‘browse code’, leading me to an earlier, archive state of the project at the time.



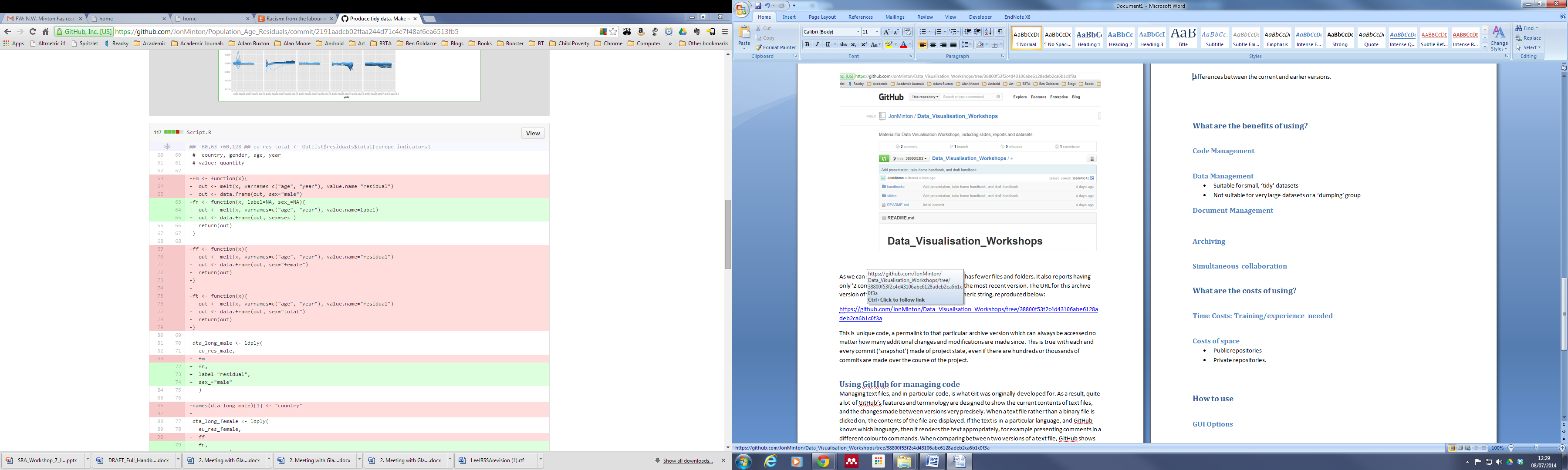
As we can see, this earlier version of the project has fewer files and folders. It also reports having only ‘2 commits’, rather than the ‘4 commits’ in the most recent version. The URL for this archive version of the project ends with a long alphanumeric string, reproduced below:

<https://github.com/JonMinton/Data_Visualisation_Workshops/tree/38800f53f2c4d43106abe6128adeb2ca6b1c0f3a>

This is unique code, a permalink to that particular archive version which can always be accessed no matter how many additional changes and modifications are made since. This is true with each and every commit (‘snapshot’) made of the project state, even if hundreds or thousands of commits are made over the course of the project.

# Using GitHub for managing code

Managing text files, and in particular code, is what Git was originally developed for. As a result, quite a lot of GitHub’s features and terminology are designed to show the current contents of text files, and the changes made between versions very precisely. When a text file rather than a binary file is clicked on, the contents of the file are displayed. If the text is in a particular language, and GitHub knows which language, then it renders the text appropriately, for example presenting comments in a different colour to commands. When comparing between two versions of a text file, GitHub shows precisely which lines have been changed since the previous version. An example of this is shown below:



Deletions are highlighted in red, and additions are highlighted in green. Line numbers are shown in two columns along the left, with the left column showing the line numbers of the deleted text, and the right column showing the line numbers of the added text.

Git version control has been closely integrated in new versions of RStudio, the integrated design environment (IDE) for R which has quickly become the standard way of managing projects in R. Although Stata and SPSS do not currently have the same level of version control integration, the code management and broader project management features shown above are ‘agnostic’ with respect to the types of text file used. They are likely to be as effective at managing Stata .do files or SPSS syntax files.

# The benefits of using Git/GitHub

There are multiple benefits to using Git/GitHub for project management, rather than something like Google Drive or Dropbox alone. This section will briefly discuss the potential uses and benefits of using Git/GitHub in the context of code management, preliminary presentation of results, data storage, and written document management.

## Git/GitHub for code management

Code is easy to view compare over time, effectively managed and literately archived. Multiple people can view, collaborate on and make suggestions about the code, identifying bugs and more effective ways of achieving project aims.

## Git/GitHub for images

The exception to the ‘GitHub is for text files’ position outlined above is that GitHub automatically renders most image file formats, including pdf, png and tiff formats. This means that, amongst other users, GitHub can be used as a simple image gallery.

## Git/GitHub for data

Because it takes multiple ‘snapshots’ of files, which involves creating additional data which describes the differences between file versions (i.e. metadata), Git/GitHub is *not* best suited to very large files, and so *should not* be used as a general data dumping ground for all of the data used within a project. Instead, it is recommended practice that Git-managed projects include a small text file called .gitignore. This file is a small text file listing which files and directories *not* to archive and upload. Typically, very large data file should be kept in a directory within the project which is listed in .gitignore file, to be used locally but not uploaded each time a commit is made.

For smaller data files, however, such as summary or other derived data, GitHub can be an effective way of storing the files and making them accessible to other project members. For example, the following links is to a repository in which I have placed number of ‘tidied’ data files, based on the Human Mortality Database (HMD), which have been arranged and derived in order to perform a range of additional analyses more easily than if I had the source data alone to work with: <https://github.com/JonMinton/Population_Age_Residuals/tree/master/Data/Tidy> - whereas in the original zipped file from HMD website each country’s population and death counts are stored in separate files, and in different directories, in here information from about 100 separate files have been arranged into a single dataset with an additional ‘country’ identifier. The data have been stored as text files using a comma-separated values (csv) format so do not require specific software or versions of specific software, and the total size of any one file is still manageable within GitHub.

## Git/GitHub for documents

As we saw before, although Git/GitHub is not designed to work with binary files such as Word or Powerpoint documents, it is surprisingly well suited to working with them effectively. The potential for Git version control amongst writing, especially collaborative writing, has not been lost on writers, and in addition to GitHub an alternative Git-based repository, PenFlip, has been created for this specific purpose: <https://www.penflip.com/> . In the case of quantitative research, it is likely that manuscripts are likely to be developed alongside the code and the results, and so it may make more sense to keep all files related to a research task – including Word files – in the same project, as part of a GitHub repository, rather than to try to manage code and writing using separate systems.

# The costs

As it is effectively a task specific programming language, learning to use Git from the command line require some initial training and effort. However, only a small number of commands will be required most of the time, and even a limited knowledge can be very helpful. A free introduction to the Git command line is available from the Codeschool website from the following URL:

<https://www.codeschool.com/courses/try-git>

A more polished and advanced, but also not free, follow-up course is also available:

<https://www.codeschool.com/courses/git-real>

Using the GitHub GUI will greatly reduce the need to learn the specific code required to use Git, although a good knowledge of the core concepts and processes at work when Git archives and manages files will be necessary in order for researchers to use Git effectively, regardless of whether they are using Git from the command line or graphically.

In terms of the cost of the repositories: on GitHub public repositories are free. Anyone can access a public repository, and download the material to use locally. This is in-line with the broad open-access/reproducible research philosophy that much software is based on, and the majority of my repositories are public access. However, where intellectual property and data security concerns are an issue, it is also possible to buy access to private repositories, which are only visible to select, invited users, and otherwise invisible. The costs of private repositories are very reasonable: less than one pound per month per repository. GitHub also offers a corporate version of their services, GitHub Enterprise.

# Conclusion

In this document I have described, in very broad terms, what Git and GitHub are, how they work, and some of the potential benefits they could offer to academic researchers engaged in quantitative social science research. I have also provided some examples of using GitHub from my own ongoing research. Specific details about how to use Git and GitHub have not been provided, but some links for further investigation have been provided.

1. <http://git-scm.com/> [↑](#footnote-ref-1)