PSI AIMS SIG

A guide to using Git and GitHub through R Studio

Introduction

In the regulatory environment, it is critical to ensure a transparent audit trail. With R code being open source, projects to develop code can be contributed to by many people, making version control essential. Git is a tool which allows users to work on the same programs at the same time. Code can be modified by any user for their own purpose or with the intention of suggesting that the updates made are merged back to the original master code. Much of the terminology is unfamiliar to statisticians, so this article is aimed to break down the barriers and help us all share and contribute to code in GitHub. GitHub is a commercial website that allows users free storage repositories, as long as the files are public.

This tutorial will take you through:

1. Setting up git, GitHub and Rstudio
2. Working with git and GitHub in Rstudio

Section 1. Setting up git, GitHub and Rstudio

First of all you need to:

- Install Git from https://git-scm.com/downloads
- Get a GitHub account by registering at https://github.com
- If you are not a frequent user of R then you may also need to:
  - Install the latest version of R
  - Install RStudio from https://rstudio.com/products/rstudio
  - Install Rtools from https://cran.rstudio.com/bin/windows/Rtools/
  - Install the package usethis by running in R: install.packages("usethis")

Historically, using Git was only through the terminal or external GUI, however Rstudio and the usethis package can enable quick and easy use of Git within Rstudio.

Setting your configuration In Rstudio.

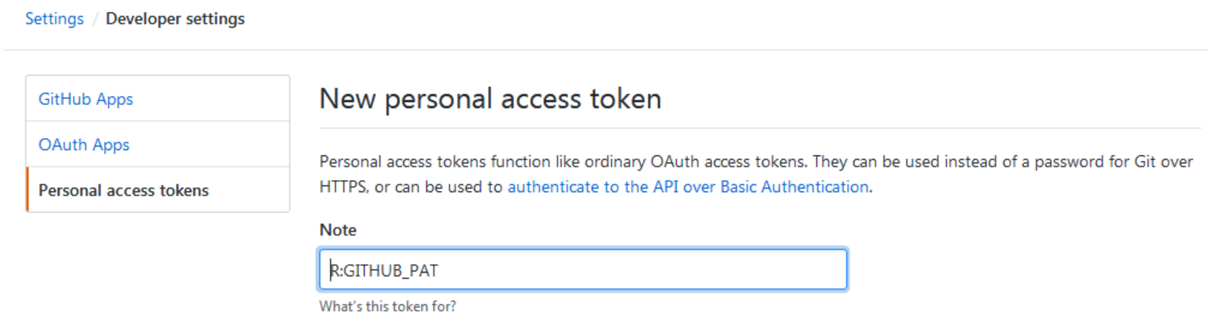
Run the following code to tell Rstudio your github username and the email you use to sign-up on GitHub. For example:

```
usethis::use_git_config
   (user.name="name.here",
    user.email ="name@gmail.com")
```

Run the following which will open a browser and take you to the github log in page.

```
usethis::browse_github_pat()
```

Log into GitHub on the browser and use it to generate a Token (default settings are OK).
Note: the that at the token is the long alphanumeric code!



New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

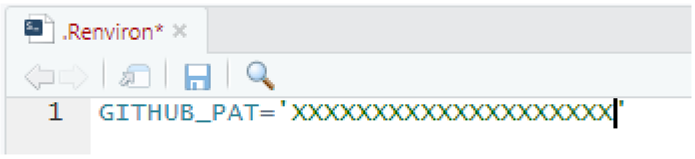
Note: R:GITHUB_PAT

What's this token for?

Go back to Rstudio and run:

```
Usethis::edit_r_environ()
```

This will open the .Renviron file. Update the file as shown below, replacing the 'XXX' with your token. Save the file.

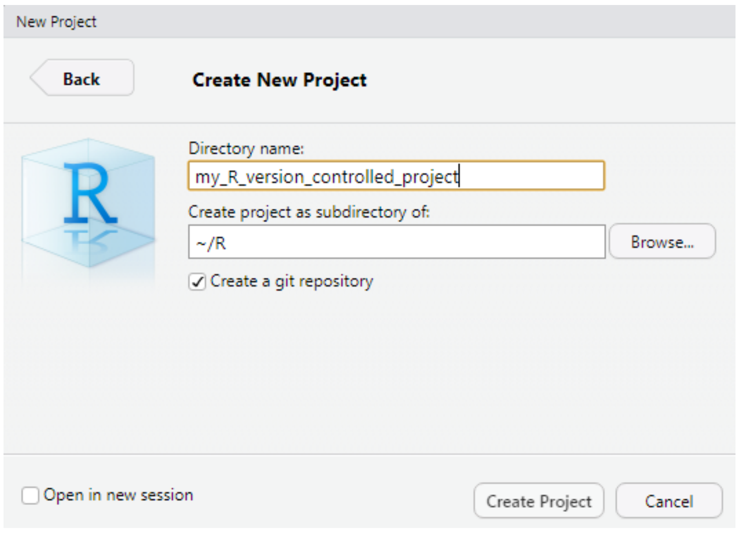


```
GITHUB_PAT='XXXXXXXXXXXXXXXXXXXXX'
```

That's it !! You are all set up to initialize your next Rstudio project using Git.

Section 2. Working with Git and GitHub in RStudio

To start a new repository in your GitHub account. In RStudio navigate to: File – New Project – Select New Directory – Create New project. Give the project a name and navigate to a subdirectory when you want to store it locally, Ensure "Create a git repository" is checked.



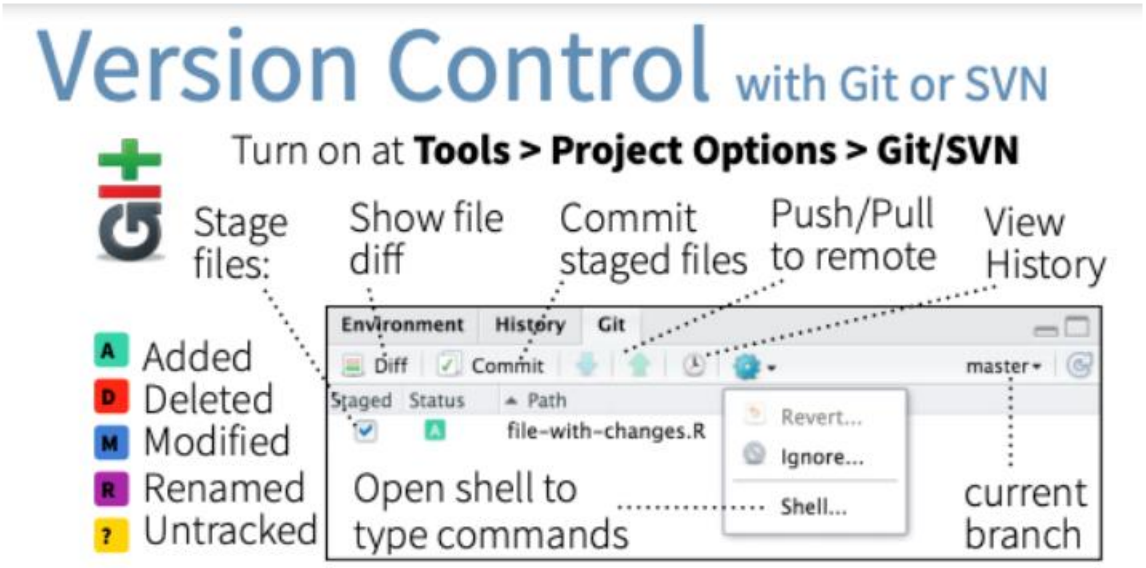
Directory name: my_R_version_controlled_project

Create project as subdirectory of: ~/R

[✓] Create a git repository

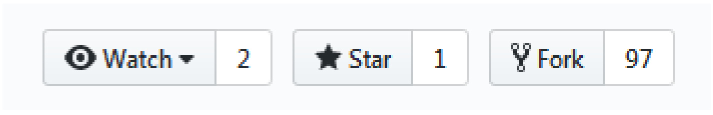
[ ] Open in new session

The Rstudio environment now includes an extra Panel "Git", which enables you to use Git in an integrated graphical user interface. Initial files are untracked and saved on your hard drive. Firstly, you have to add the file to Git to tell it you want to version control the file – this is called a "commit". As you commit you should add a comment which explains the changes made and Git will track these versions. However, this is still on your hard drive. In order to load to GitHub, you have to "Push" (Green up arrow) to upload your committed modifications back to your GitHub repository. The files are then viewable on your GitHub account and are version controlled. As you add, delete, modify or rename using Git, the status changes in line with the boxes shown below.

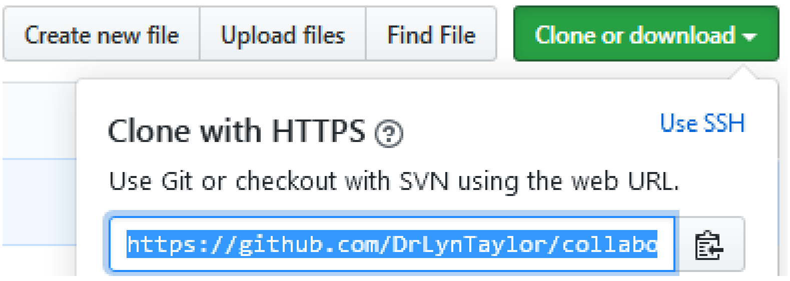


If you want to contribute to a shared project or adapt/improve code from an existing project, instead of working on a new repository of your own, you would need to clone an existing Git repository. This is often called making a "Fork" (if you don't intend to merge back to the main repository later) or a "Branch" (if you intend to work testing out changes and later merge back to the master repository at a later date).

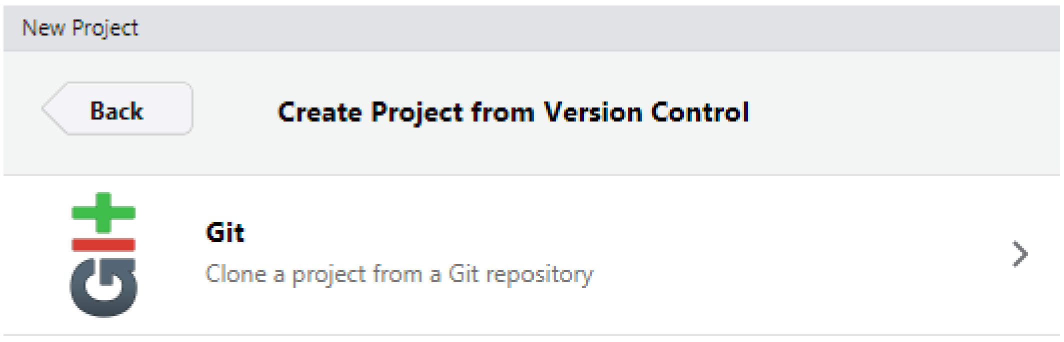
To do this, go into GitHub and navigate to the Repository you want to contribute to. Select "Fork" to make your own copy of the repository. GitHub makes a copy into your GitHub repository.



Now to be able to edit in Rstudio – select the "Clone or download" and copy the URL.


Create new file
Upload files
Find File
Clone or download
Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/DrLynTaylor/collabo

This time when you are in Rstudio: navigate to: File – New Project – Select Version Control – Git – and paste the URL into the "Clone Git Repository" – "Repository URL" field. Then create project.


New Project
Back
Create Project from Version Control
Git
Clone a project from a Git repository

Now you will see all the files associated with the project in the Files section on the Git Tab in RStudio. You can edit and save the R code making any changes required. When you save, your new file will appear in the Git window but this is only storing the file locally. To save it as a version you have to commit it to Git. It will ask you to write a "commit message" to document the changes you have made. To save a committed version to GitHub, you have to "push" it back up to the repository (by clicking the up green arrow). This is now stored in the forked repository that you cloned.

Suppose, you believe your edits to be superior to the original master file and would like the author to accept your changes into the master. This is called branching and you need to submit a "Pull" request (by clicking the down blue arrow), which means Git compares your branch with the master and merges your changes into the master (if approved by the author). GitHub checks your requested merge wouldn't create any conflicts and confirms you are able to merge. Code authors should write atomic code (i.e. code that doesn't break by removing bits that were committed at different times) to ensure collaborations are successful. Any future forks of the master GitHub repository, would be from this new master version. When submitting a pull, you should include an informative message of the changes you are suggesting, to assist the author to decide whether to accept your changes or not. In other words, a "Pull" request asks the maintainer of the repository to pull your changes into their repository.

Once your changes are uploaded, you will also need to "Pull" any other changes that occur in the master back to your own repository if you want to keep contributing to the project.

Summary

This short tutorial is intended to get you started using Git, GitHub and RStudio to collaborate on producing R code. It was put together by Lyn Taylor (PHASTAR), based on a Sheffield R Users Group presentation by Mark Dunning (Sheffield Bioinformatics Core Director, sbc.shef.ac.uk) and Anna Krystalli of the University of Sheffield.

Further information can be obtained at: https://git-scm.com/book/en/v2 and https://r-bio.github.io/intro-git-rstudio/