Git, GitHub, and R:
R User Group

Chris Greyson-Gaito

Prerequisites

- Install Git (Windows, Mac, Linux)
- Register for GitHub

Benefits of Git and GitHub

- **Git**
  - Version Control your R scripts and your manuscripts
    - All of your edits are saved and so you can go back to any edit that you have made
    - Collaborating is very easy with git and GitHub and all of the edits made by contributors can be seen
    - You can tag different versions and easily go back to a previous version
    - You can experiment with your coding without worrying about breaking the original code
- **GitHub**
  - Publish your R scripts for your published papers (see Monica Granados R script for published paper for an example)
  - Collaboration
    - If there is useful code already written, you can copy it using GitHub and use it (as long as you cite it)
    - Can collaborate with fellow scientists on R scripts or on manuscripts (doable with .docx, easier with .tex/.md)

Basics of Git (Conceptual)

To explore Git we will go through TryGit together.

Other useful webpages on how git works:

`http://r-bio.github.io/intro-git-rstudio/`

`https://www.git-tower.com/blog/workflow-of-version-control`

`http://nyuccl.org/pages/gittutorial/`

Using Rstudio with Git

1. If you haven't already, install Git
2. In RStudio, click on Tools → Global Options → Git/SVN
3. Check the "Enable version control interface for RStudio projects" button
4. Ensure that the Git executable box contains the correct path to the Git executable on your computer and ensure the box "Use Git Bash as shell for Git projects" is checked.
5. Click on Create RSA key (we will return to this key later). Now close the Options box.
6. Create a project (with the following folders: R/ figs/ doc/ data/). Also create a new R script in the R folder of the project.
7. In Project Options → Git/SVN → select Git in the Version Control system (click yes to both)
8. Edit the R script and save the file
9. Create a textfile (in RStudio) and add the following text → figs/ doc/ data/ . Save this file as .gitignore (solely .gitignore, no text in front of the .) in your project folder (i.e in the same place as the .proj file).
10. Click on the Git button and then click on Commit
11. Check the boxes under "Staged". This is the same as "git add filename".
12. Write a message in the "Commit message" box and then click on the Commit button. This is the same as "git -m "commit message" ".

Useful Websites:
`https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Gi`
`https://support.rstudio.com/hc/en-us/articles/200526207`
`https://jennybc.github.io/2014-05-12-ubc/ubc-r/session03_git.html`

Collaborative R Scripts using GitHub

There are two methods for collaboration in GitHub (forking/issue pull request method and the add collaborator method). In this tutorial we will use the add collaborator method.

1. If you haven't already sign up to GitHub
2. Remember that SSH RSA key, we need to use this for GitHub. In RStudio, click on Tools → Global Options → Git/SVN → "View public key"
3. Copy the text of the public key.
4. In your GitHub login, click on the top right menu button and then click on "Settings". Now click on "SSH and GPG keys". Click on "New SSH key" and paste your public key (what you just copied) into the "Key" box and give it a title. Click on "Add SSH key".
5. Give Chris your GitHub username and he will invite you to collaborate on a project
6. After accepting the invite, open the RUserGroup_Git repository and click on the "Clone or download" button.
7. Copy the text that appears (i.e. gitgithub.com:cgreysongaito/RUserGroup_Git.git).

8. In RStudio, click on New Project → Version Control → Git
9. Paste the text you just copied into "Repository URL" and give the Project directory name as RUserGroup_Git. For the third box, place this repository in a location that makes sense for you.
10. Open R/CollabScript.r, add some R code (anything) and save the r script.
11. Now, add and commit your changes (the same method as above).
12. Click on the Git button and click on "Push branch" and enter your password for your SSH key.
13. If someone else has made a change to the R script and you want the most recent script on your computer, instead click on "Pull branch".

For the fork/issue pull request, the basic idea is that you put a local version of someone's project onto your GitHub system (this is forking, but any changes you make to this local version can not be pushed to the original project). If you want the original project to have your changes you issue a pull request and the original project owner decides whether or not to incorporate your changes. For more information see Git Fork and RStudio Git Intro.

R scripts on GitHub for published papers (your homework :))

Before you send your manuscript to a journal, your r script should be published. The great thing about git and GitHub is that if your reviewers want changes to your analysis, you can change your r script and republish it and people can see those changes (open and transparent science!).

1. When your r script is ready (and you have just done your final commit to your local repository), login to GitHub and create a new repository (by clicking on Repositories and New).
2. Copy the url of the webpage of your repository (e.g https://github.com/cgreysongaito/RUserGroup_Git)
3. In RStudio, click on Tools → Shell and type in the Shell git remote add origin <url you just copied (without the <>)> and press enter
4. Back in GitHub, click on "Clone or download" and copy the text.
5. Now in the RStudio shell type git config remote.origin.url <text you just copied (without the <>)> and press enter
6. Now push your repository to GitHub.

You have now published your R script. A nice feature of git is to tag a commit with a version number. This is useful for reviewers and other people to see where your different versions are (and in your citation for the published r script you can give a version number).

To add a version number:

1. Open the R project that you want to publish
2. Click on Tools → Shell and type git tag -a <version number> here -m "any message here"
3. Push your repository to GitHub using RStudio
4. Unfortunately, RStudio does not automatically push tags. Therefore, in the shell type git push origin −−tags

More information on tagging can be found here.

Other useful links for RStudio and GitHub:
`https://www.r-bloggers.com/rstudio-pushing-to-github-with-ssh-authentication/`
`https://www.r-bloggers.com/rstudio-and-github/`

Licenses

It is very important that you include a license for anything you publish on GitHub (to protect yourself and to help people understand what they can and not use). Your best bet is probably to use a Creative Commons Attribution Share-Alike license.

Here are a few links to help you choose a license and to put the license into your GitHub repository.

`https://choosealicense.com/`
`https://help.github.com/articles/licensing-a-repository/`
`https://creativecommons.org/licenses/`